Research Question

The research question I want to explore is: What are the key causes or correlations associated with Anxiety in patients? Given the prevalence ofanxiety among Americans, it's essential for hospitals and healthcare providers to develop effective treatment strategies. Comprehensive treatment depends on understanding the relationships—if not causal links—between various health conditions and data points. To enhance the predictive accuracy, I plan to use K-Nearest Neighbors classification to develop a more effective model.

Objectives and Goals of Analysis

I wish to look at the possibility of using KNN classification to produce a more beneficial model. My research from the second performance assessment in D208 will be expanded upon in this study to see whether KNN classification can increase model accuracy. For patients and hospitals, it is important to understand the links between various illnesses, even if they are not directly causal. This information lowers the chance of developing new problems, helps manage existing health difficulties, and offers insightful information about an individual's healthcare status. Hospitals can lower rehospitalization rates, avoid extended stays, and enhance the quality of care they provide by successfully addressing these problems.

Justification of Classification Model

My area of interest is the K-Nearest Neighbors (KNN) classification technique and the research questions it raises. With this method, data points are given labels according to their characteristics. KNN looks at a predetermined number ('k') of the labeled data points that are the closest to a newly discovered unlabeled data point. In a sense, these nearby locations "vote" on the label for the unidentified data point. Every unlabeled data point is assigned a predicted label using this voting procedure, which is based on patterns seen in the labeled dataset.

While the value of 'k' can be changed to optimize the model, once selected, it cannot be changed for that specific model. The Euclidean distance, which computes the straight-line distance between the new data point and other points in the dataset, is usually used to determine which neighbors are the closest. Nevertheless, as stated in the scikit-learn documentation for KNeighborsClassifier, alternative distance metrics can be applied. The expected outcome of this analysis is that the model will predict with a classification rate above 50% on the test data.

B2: Assumptions of a KNN Classification Model

No matter the scenario or the item being classified, the core idea behind any KNN model is that similar objects are located near each other on a graph. This is because the model relies on nearby datapoints to determine their labels, helping to identify the unknown datapoint. If this isn't true, the neighboring points won't be effective in classifying the unknown datapoint, causing the KNN model to fail to perform correctly.

Benefits of Choosing Tools

Python will be my tool of choice for this analysis job. Because of its specific packages, Python is a programming language that is excellent at supporting data science jobs. It's also the only programming language that I am proficient in using. I'll be using a few different Python libraries for this analysis:

**pandas**: Facilitates handling the dataset in a table-like format.

* **pandas' CategoricalDtype**: Allows creation of categorical and ordinal columns.

**NumPy**: Enables various mathematical operations and value assignments within the dataset.

**Seaborn and Matplotlib**: Provide graphing and visualization capabilities.

**SciPy's statsmodels API**: Offers several critical functions for our analysis.

* **Variance Inflation Factor (VIF) function**: Helps calculate VIF to check for multicollinearity among features.
* **Mosaic function**: Generates mosaic plots for visualizing bivariate relationships in categorical data.

**sklearn**: Supplies essential functions for this analysis.

* **train\_test\_split**: Easily splits the dataset into training and testing sets.
* **preprocessing module**: Contains various functions for preprocessing our data.
* **SelectKBest and f\_classif**: Help identify the top features out of more than 20 potential candidates for the model.
* **KNeighborsClassifier**: The main classifier for our data.
* **GridSearchCV**: Determines the optimal value of 'k' for the KNeighborsClassifier.
* **confusion\_matrix**: Prints a confusion matrix for the KNN model.
* **roc\_auc\_score**: Computes the Area Under the Curve (AUC) score for the KNN model.
* **roc\_curve**: Plots the Receiver Operating Characteristic (ROC) curve of the KNN model.
* **classification\_report**: Summarizes the metrics for the KNN model.

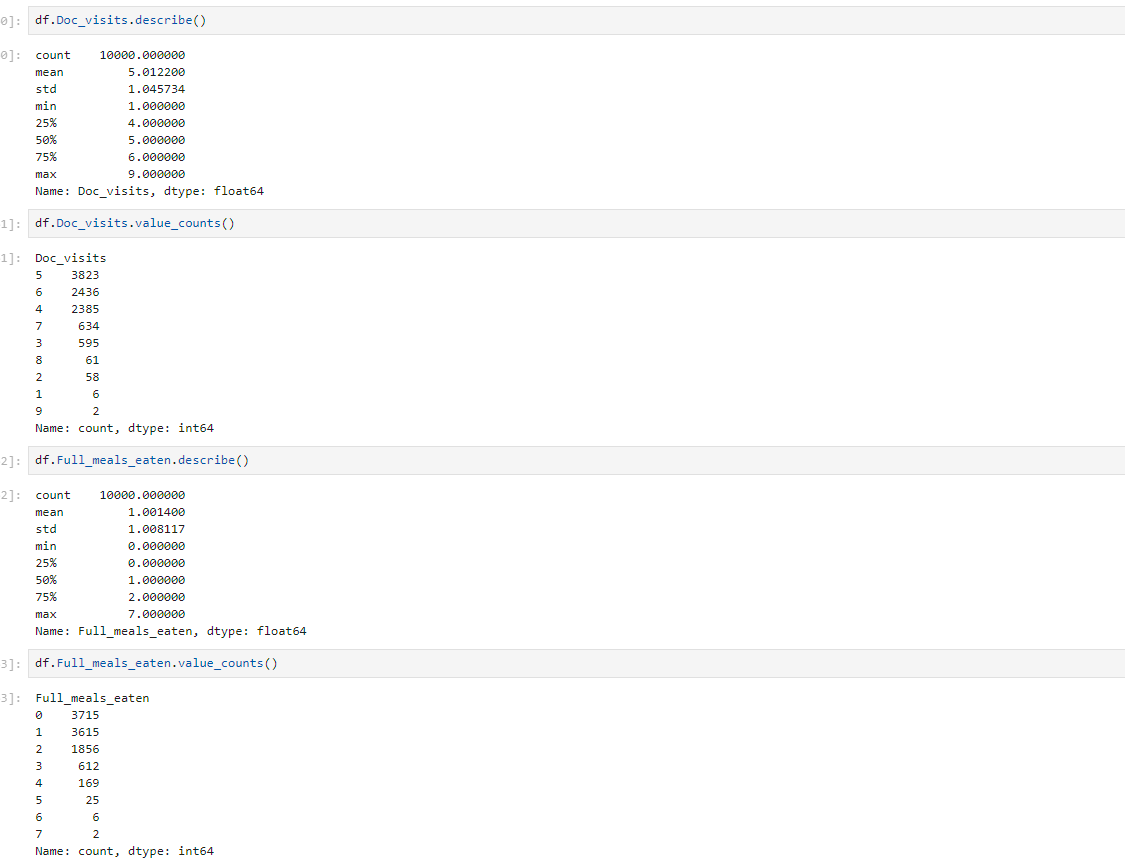
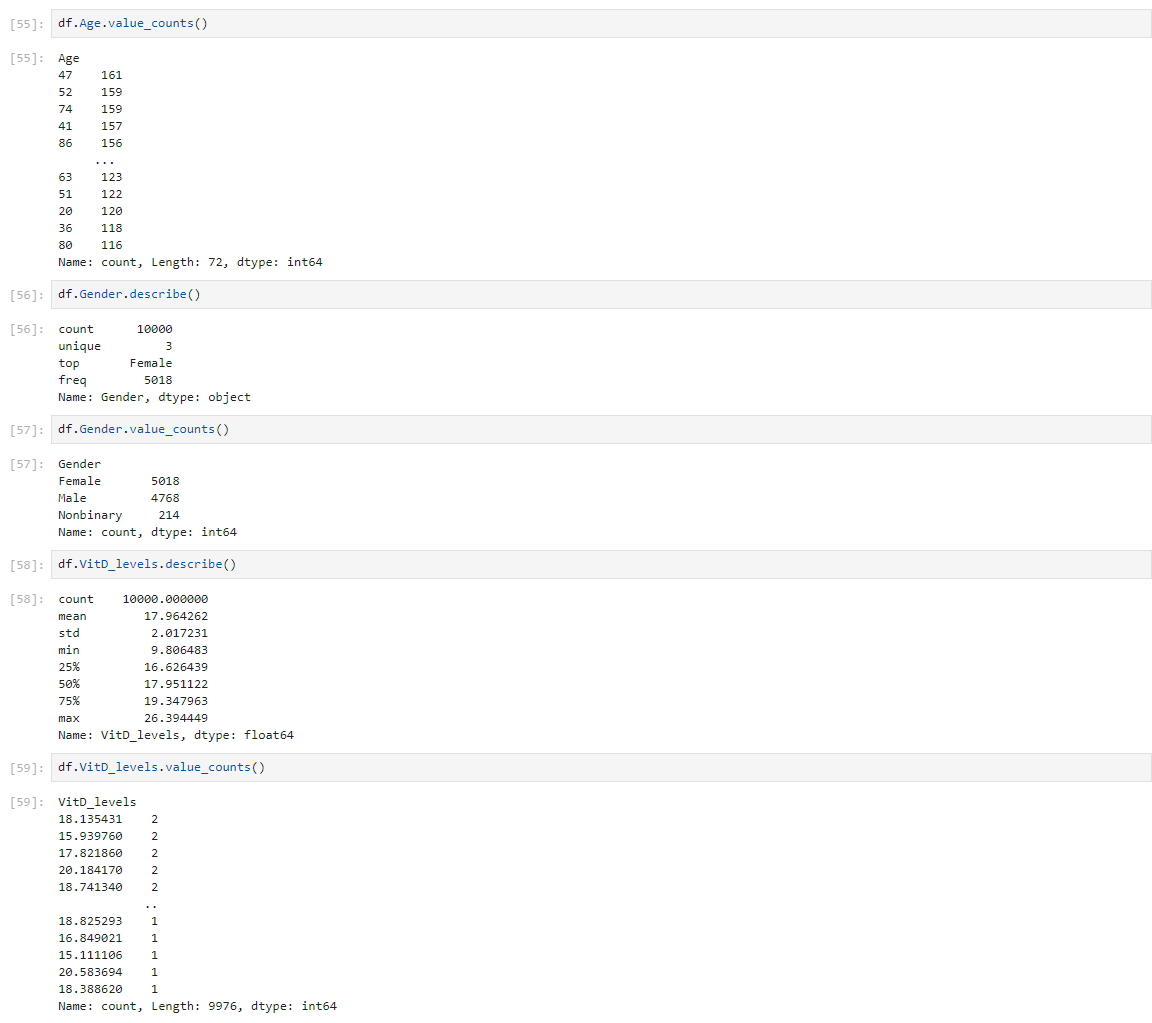
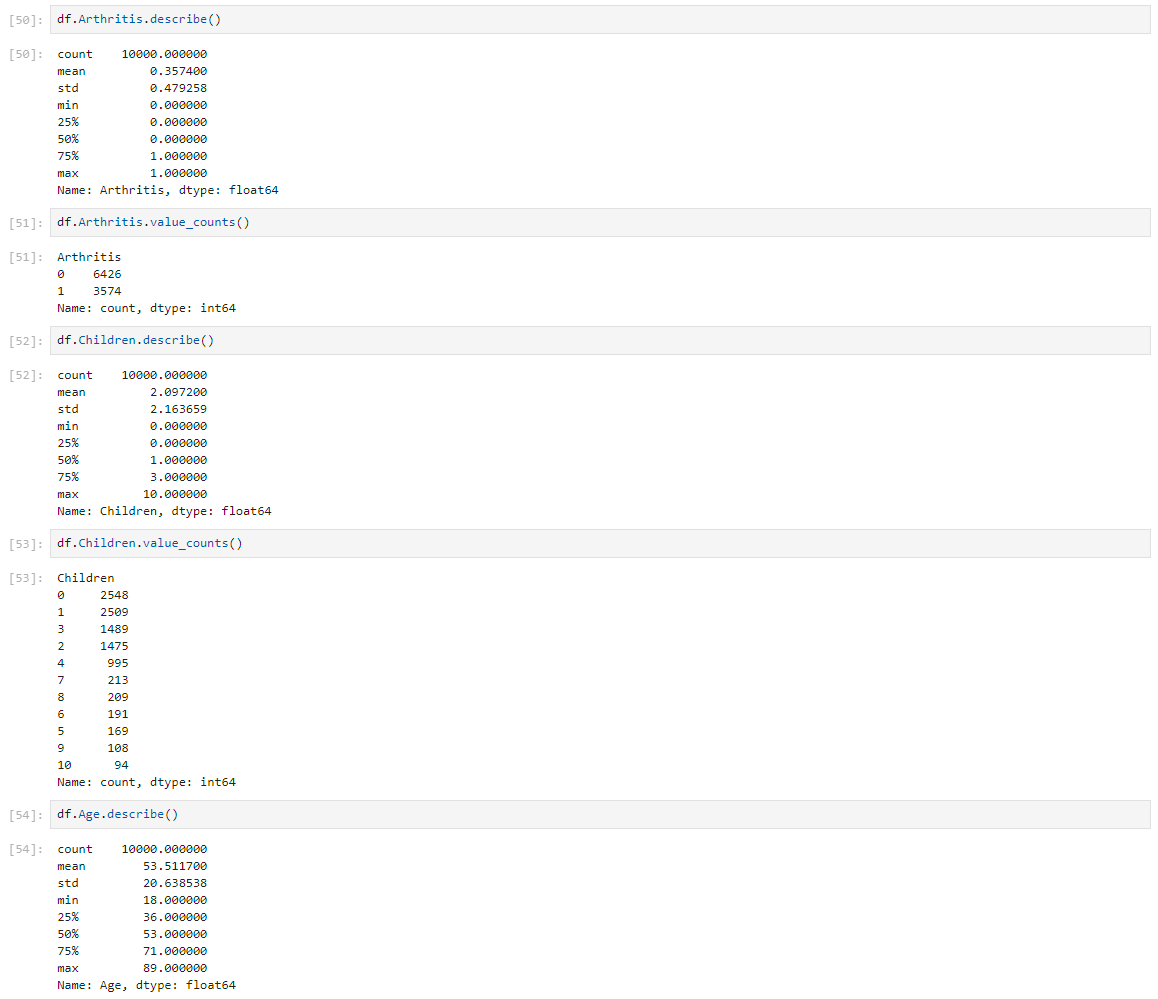
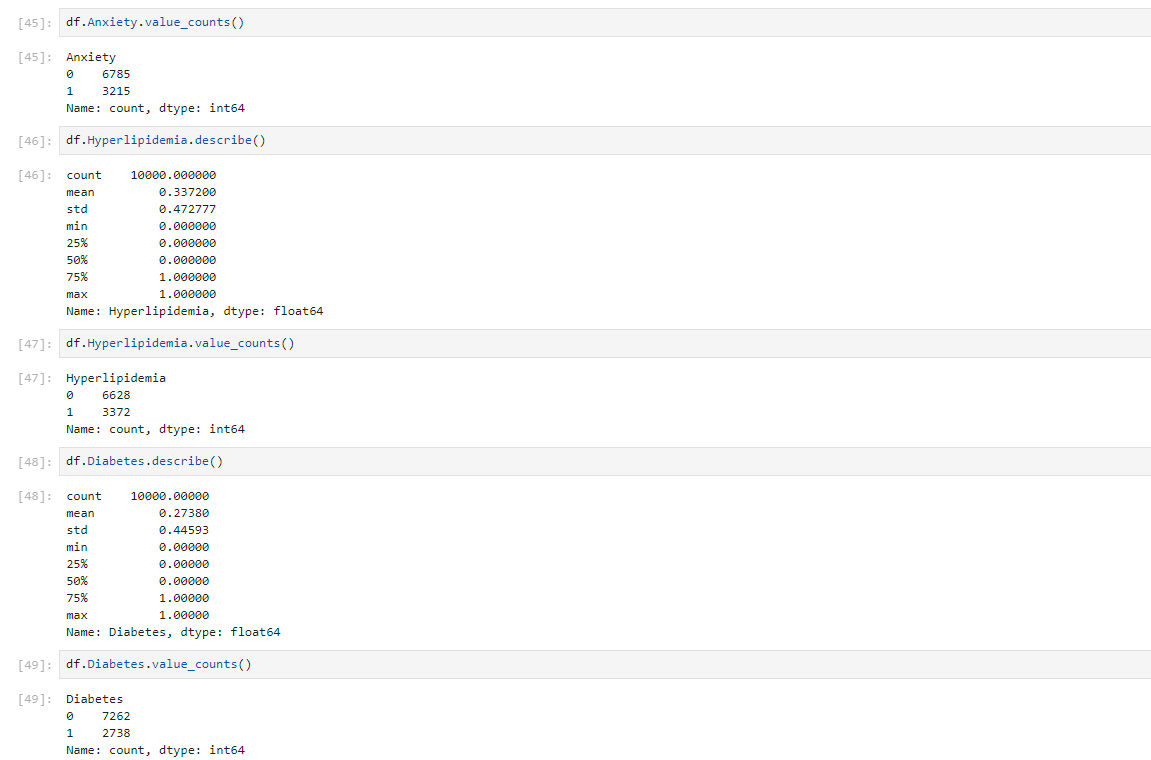
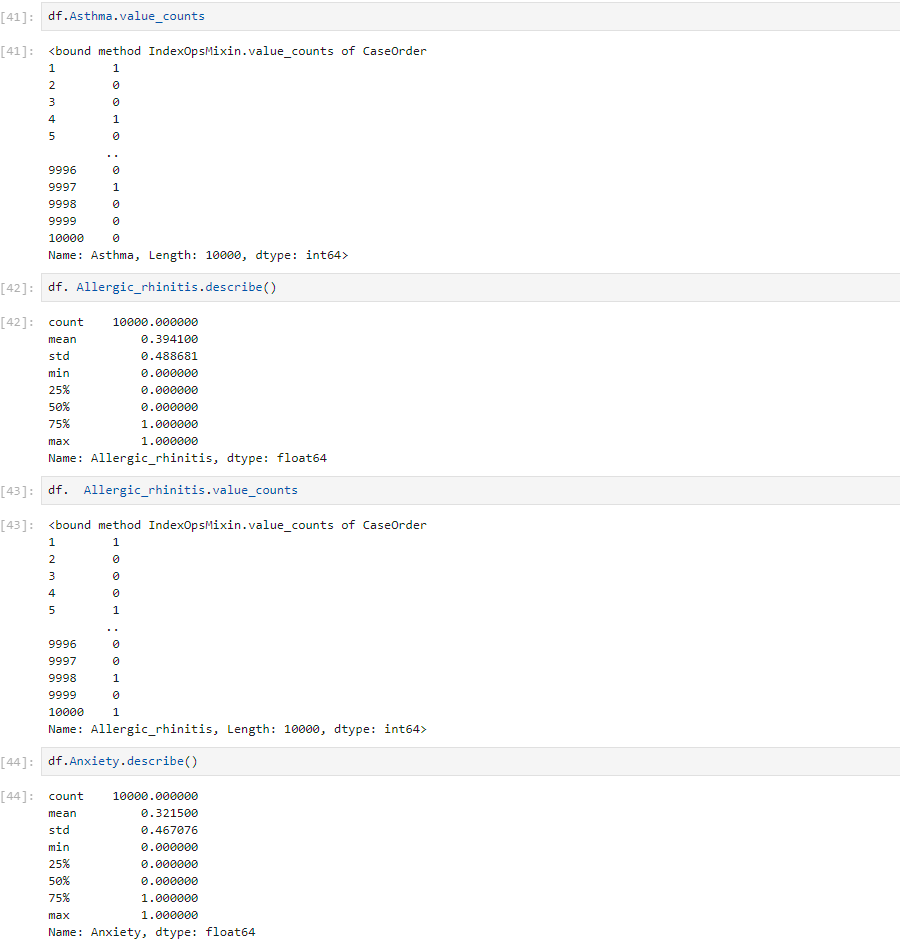
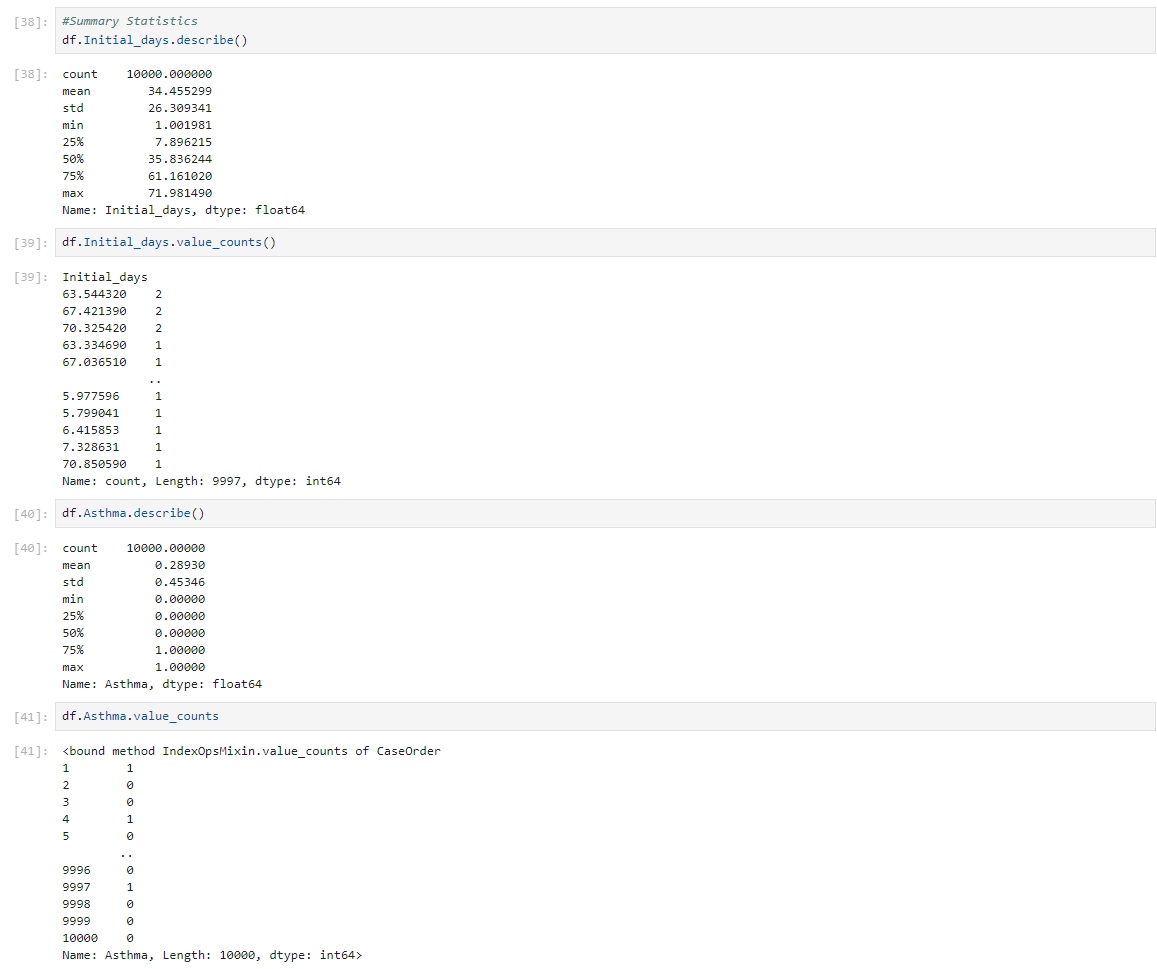


C1: Data Preparation Goals and Necessary Manipulation

There are several data preparation and cleaning steps we need to perform, and one crucial step for KNN classification is one hot encoding. One hot encoding converts categorical data into a numerical format by creating separate columns for each category, using 1s and 0s. For example, in a gender column with values like Female, Male, and Nonbinary, we can create two new columns. If the first column has a 1, it indicates the patient is male. If the second column has a 1, it indicates the patient is nonbinary. If both columns have 0s, the patient is female. This can be easily done with pandas' get\_dummies() function. By representing the data numerically, our KNN classifier can effectively process it, as it wouldn't understand categorical labels like "male" otherwise.

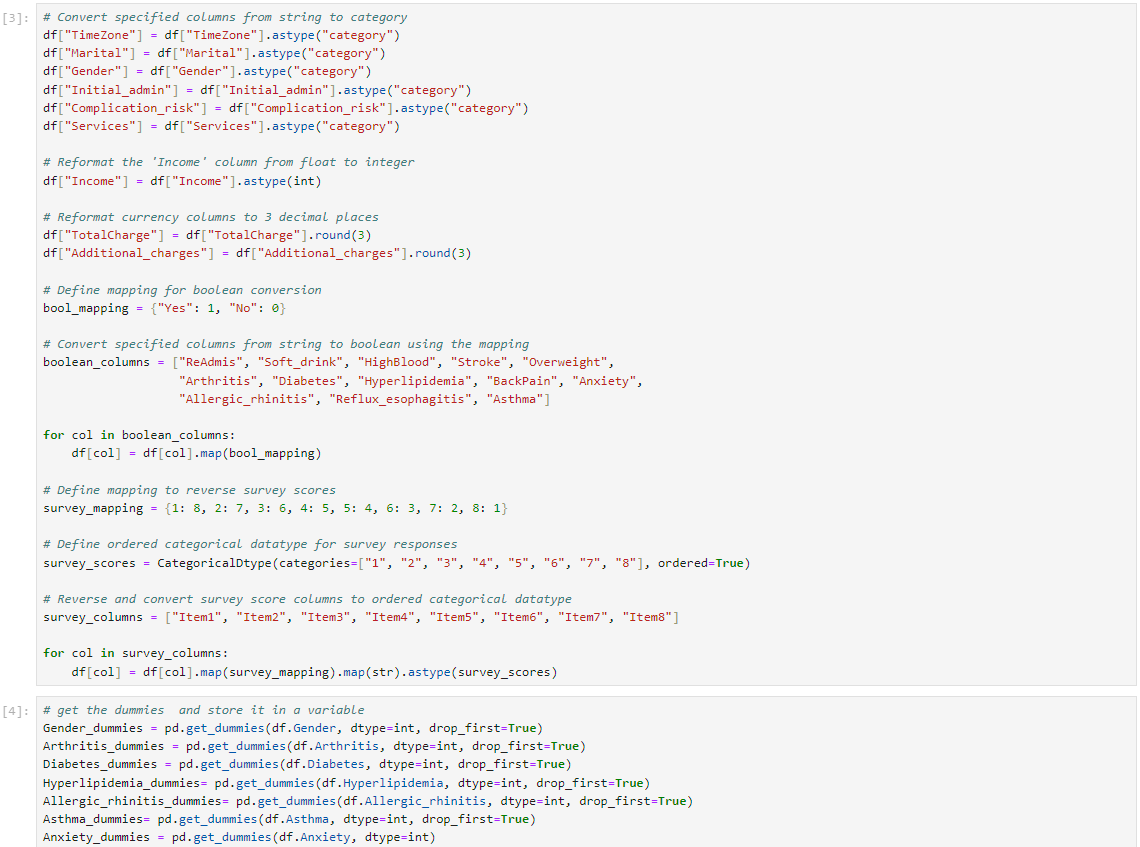
**C2 Variable Selection and Identification**

* Numerical: Number of Children
* Numerical: Age
* Numericall: Gender
* Numerical: Vitamin D levels
* Numerical: Number of Dr Visits
* Numerical: Full meals
* Categorical: Categorical: Arthritis
* Categorical: Diabetes
* Categorical: Hyperlipidemia
* Categorical: Anxiety
* Categorical: Allergic Rhinitis
* Categorical: Asthma
* Numerical: Initial Days

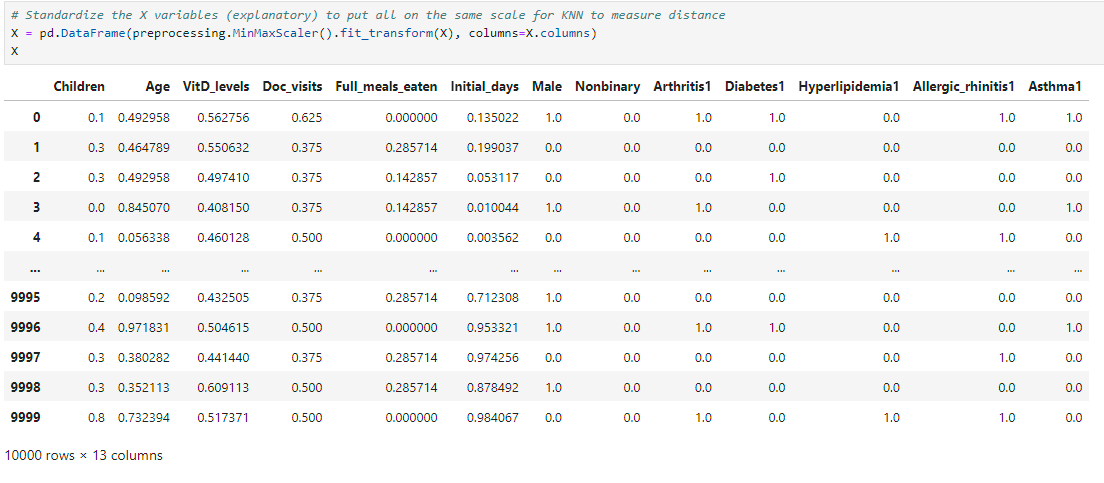


C3: Preparation of Data

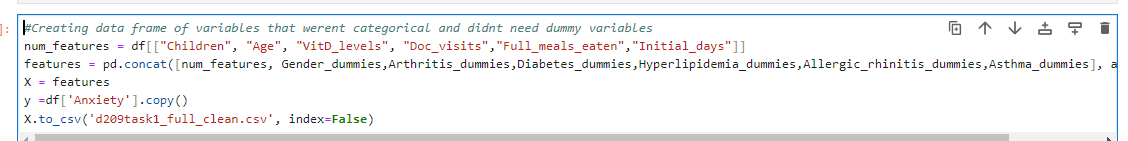
I'm going to clean up the data using the same adf.pproach I applied in D208. This includes handling the ordinal survey responses and converting binary categorical values from Yes/No or True/False to 1/0.Once these columns are corrected, I'll explore the data with functions like describe() and value\_counts() to ensure it's ready for classification analysis. I've already checked for nulls using info(). This step isn't required by the rubric, but it's recommended, so I'll do it quickly. After confirming the data is suitable, I'll generate dummy columns for categorical variables and insert them into the dataframe in place of the original columns, maintaining the order I'm accustomed to from previous projects. Then, I'll remove any columns that aren't needed. I'll double-check the dataframe to make sure everything looks right after these changes.

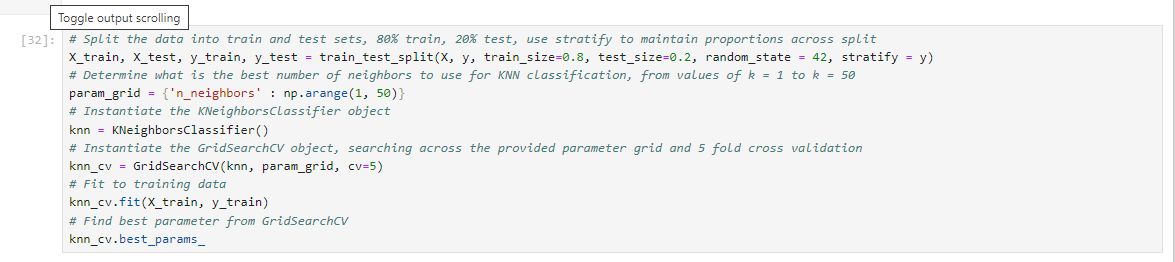


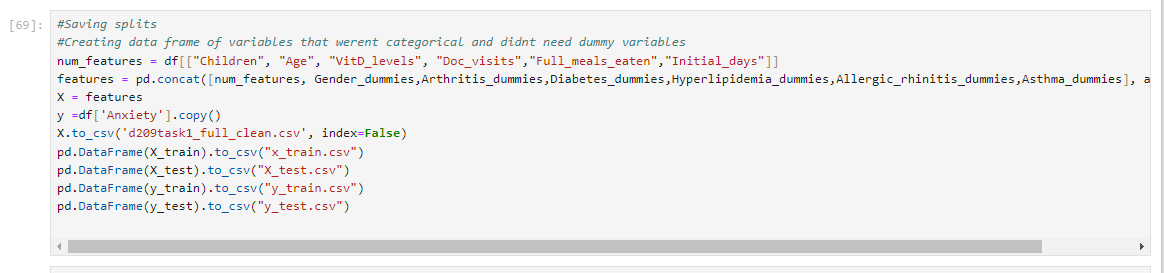
Next, I'll rescale the feature set using sklearn's MinMaxScaler(). This transformation adjusts data to a range of 0 to 1 by subtracting the column's minimum value from each entry and dividing by the range (max-min). This ensures all features are on the same scale, preventing any from dominating the model due to larger values.

.

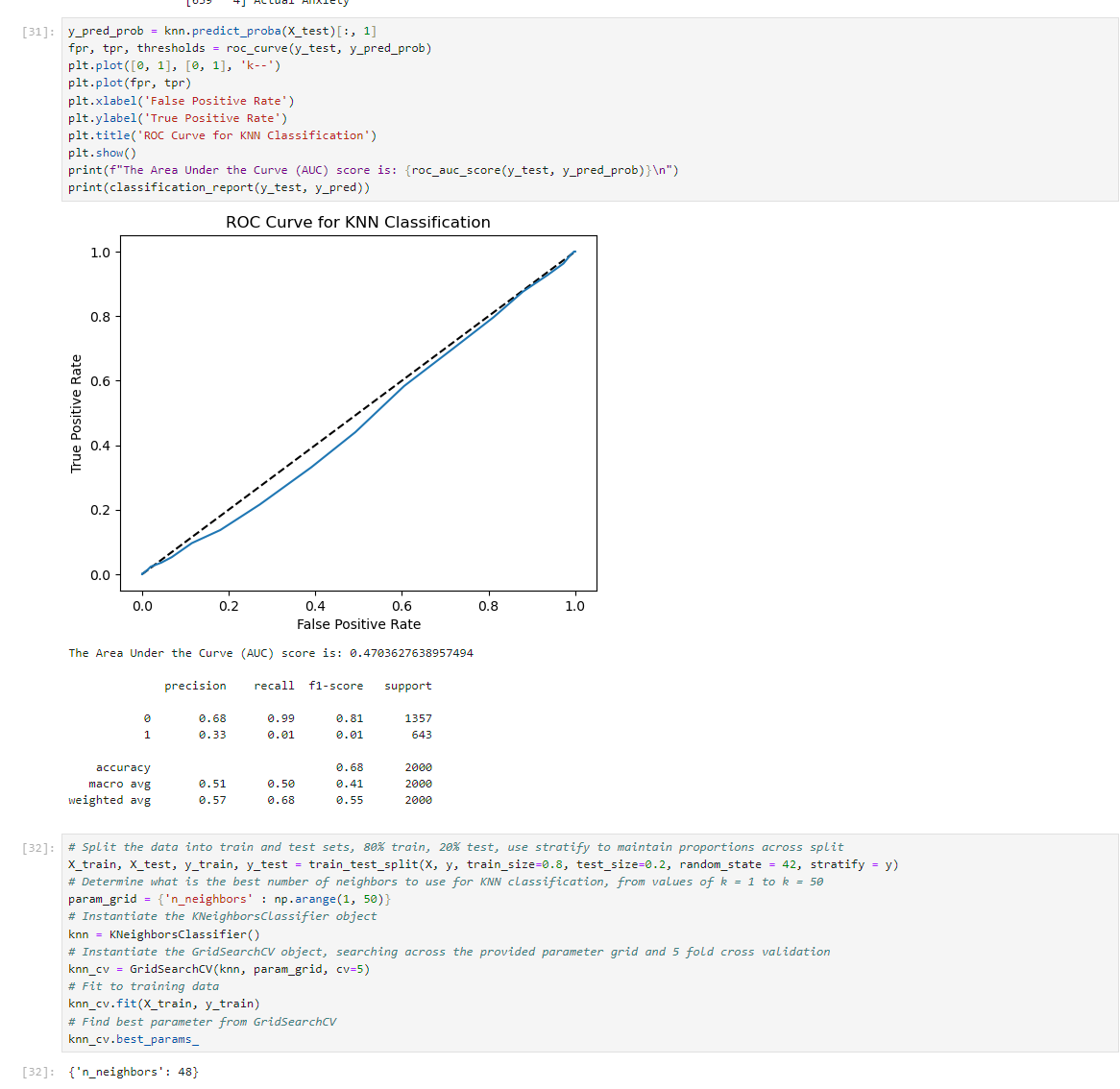
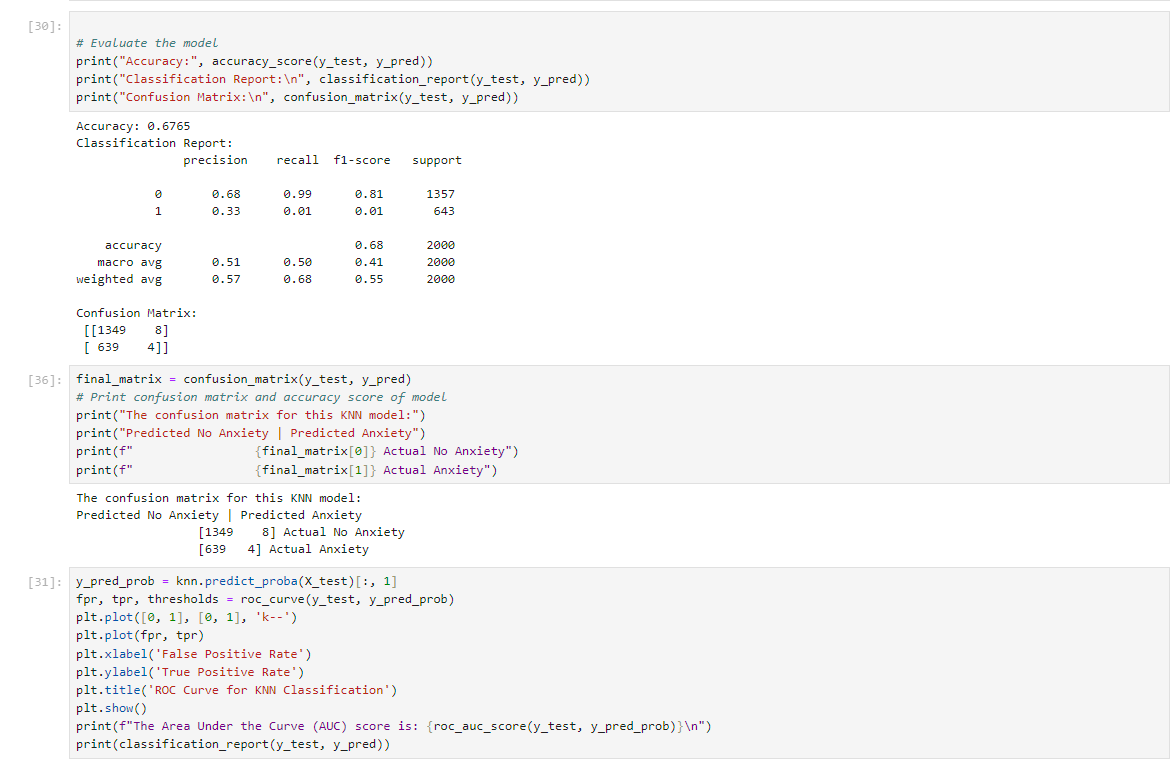
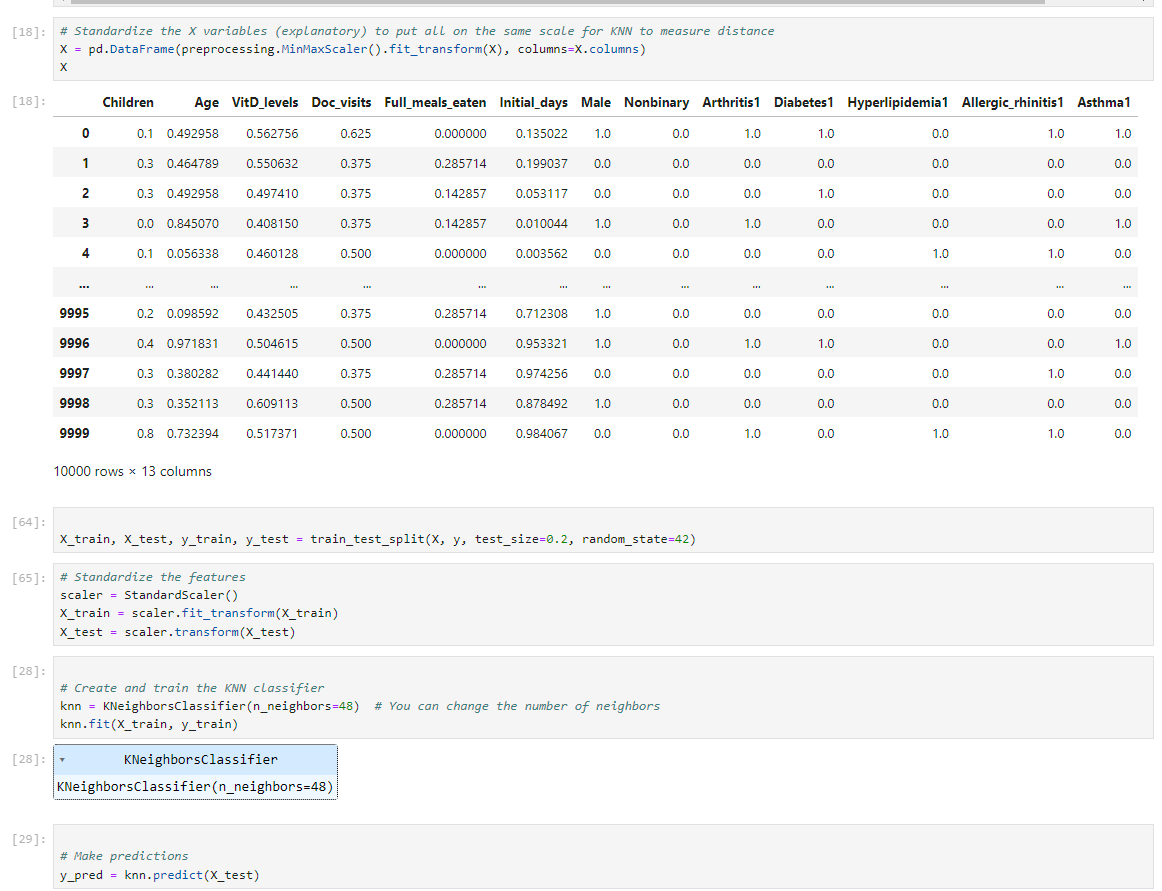
*C4: :Below Is a copy of my prepared data set*

**

*D1 Data Splitting*

**

*D2 Analysis*

**

To perform a K-Nearest Neighbors (KNN) classification, I need to determine the right value for 'k'. Using Hugo Bowne-Anderson's DataCamp lesson on hyperparameter tuning, I can leverage GridSearchCV to test various 'k' values and identify the one that provides the best accuracy. Once the optimal 'k' is found, I can proceed with the KNN classification. The rubric requires the Area Under the Curve (AUC) score for the classification model, so I calculated this using the example code on AUC computation from the same DataCamp material. Additionally, I'll include the ROC curve, generated with the provided code, to visually represent the model's performance.

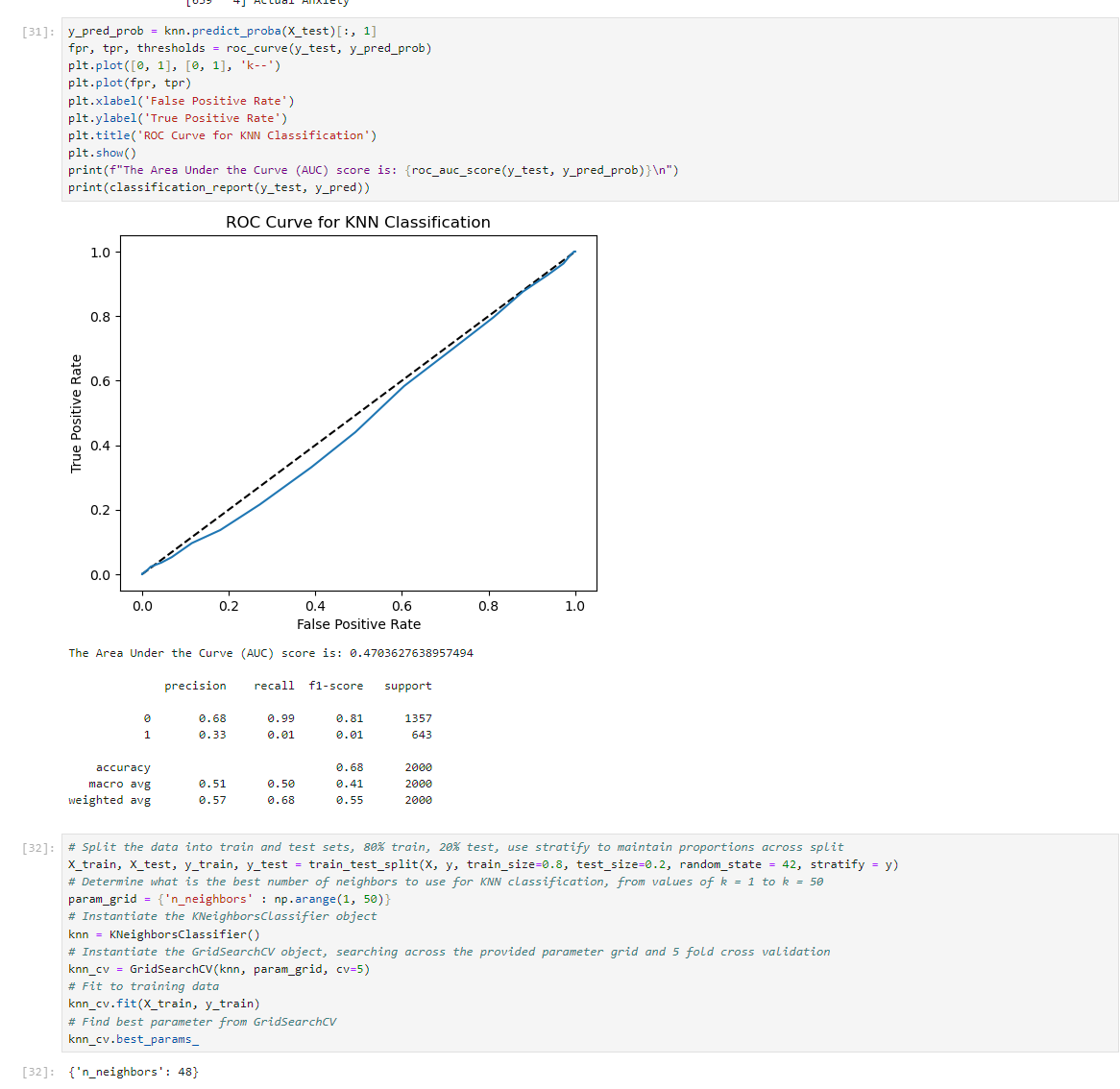
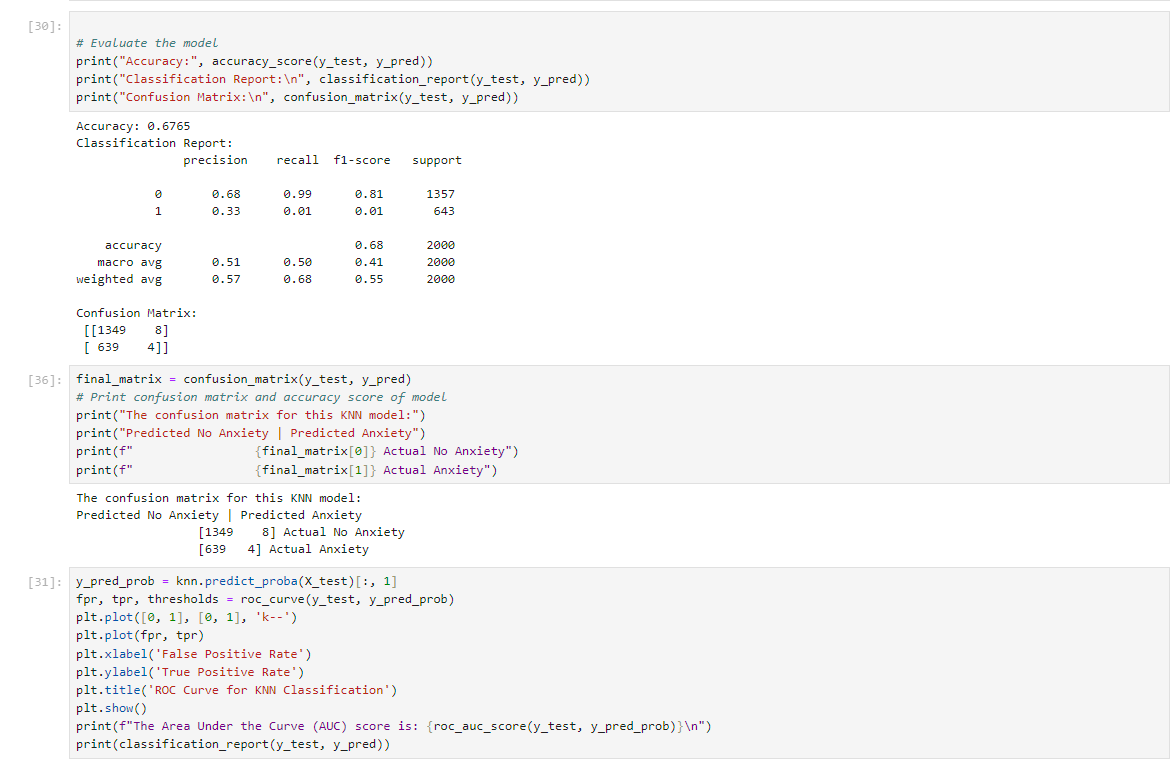
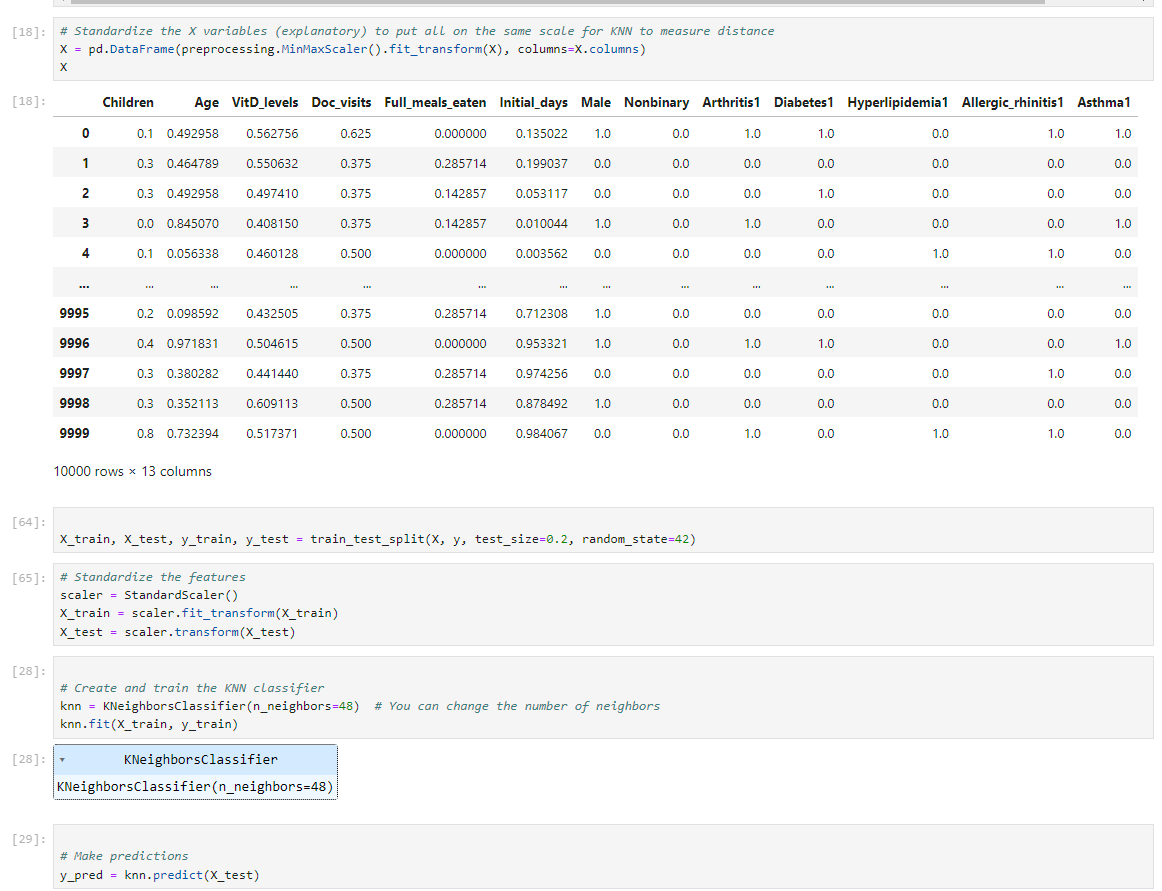
As explained earlier, the K-Nearest Neighbors classification assigns labels to data points based on their features by examining the labels of the nearest points. The 'K' in KNN denotes the number of neighbors considered and is set when the classifier is instantiated. To find the best 'k' value, I used hyperparameter tuning instead of picking one arbitrarily. This involved using GridSearchCV to test a range of 'k' values and identify the most effective one.

Testing values from 1 to 50, the process revealed that k = 48 was the best choice. Using this value, I built the KNN classifier and fitted it to the training data. I then made predictions on the test data and created a confusion matrix. Out of 2,000 test set observations, 1,353 were correctly classified, while 747 were not, resulting in an accuracy of 67.65%.

To illustrate the model's performance, I plotted a receiver operating characteristic (ROC) curve. This graph shows the KNN classification model's performance, with the diagonal line in the middle representing random classification at 50% accuracy. A curve above this line indicates better-than-random performance, while a curve below indicates worse performance. The curve for this model performs slightly worse than random classification, as indicated by its Area Under the Curve (AUC) score of 0.4703.

The KNN classification allows you to provide different weights to your neighbors. Although you can select to weight neighbors according to the inverse of their distance or even apply a custom function, by default, all neighbors are given equal weight. The classification model used uniform weighting and the default value in this instance. Furthermore, KNN supports a variety of techniques for determining the separation between the closest neighbors. The standard brute-force approach utilizing Euclidean distance was used for this categorization.

D3 Classification Code



E1 Accuracy

The accuracy of this KNN classification model was a little above 67%. However, the fact that the majority of the data (67.5%) includes instances with no anxiety (0) has a significant impact on its accuracy. The model is not very accurate, especially when it comes to accurately recognizing the times when the patient actually has anxiety (shown as 1).

A 50% correct classification rate is indicated by the central diagonal line in the ROC map, which is consistent with a totally random categorization. The performance of a classifier is represented on this graph: good predictions (up to 100% correct in the top left) are indicated by a curve above the diagonal, whereas poor predictions are indicated by a curve below the diagonal. A flawless model receives a score of 1.0, while a completely erroneous model receives a score of 0.0. The Area Under the Curve (AUC) score indicates the percentage of the graph's area under the model's curve. The AUC for this model is 0.4703, which is less than 50%. This score shows that, despite the KNN model's apparent higher accuracy, its performance is marginally poorer than random categorization, fitting the curve that is depicted just under 50%.

E2 Model Results

According to the model's AUC score, random guessing is just slightly more efficient than the model's performance. In particular, its low precision indicates that it has difficulty correctly predicting anxiety patients. With each case assigned a projected probability of 0, the model's performance would fall exactly on the diagonal, yielding an AUC score of 0.5 if it predicted that no patients had anxiety. This emphasizes how ineffective the existing classification model is.

E3 Classification Limits

A machine learning (ML) model operates on the principle that similar items are located near each other when visualized on a graph. The KNN classification method relies heavily on this principle, assuming that the 'neighbors' of a data point share the same label. This is essential for the method to work effectively. However, the low performance of this model suggests that the instances of Anxiety in the medical dataset are distributed too randomly for neighboring data points to be meaningful. As a result, the KNN classification method is ineffective in this context, limiting its usefulness.

E4 Recommended Action

Age and the first few days are the two variables that most likely have the greatest statistical significance, with children potentially standing in for age. My data, however, point to a negligible correlation between these factors and anxiety.

Anxiety in patients might be more accurately predicted using a dataset that is more closely linked to healthcare data, such as treatment kinds, beginning complaints, patient weight, activity levels, and so on. In order to safeguard patient privacy, I advise the hospital system to increase the amount of this kind of data it collects and to make sure it is anonymized.

Source and Code References

I used code from William Townsend's D208 Task 2 Performance Assessment Submission to clean and prepare the dataset.

Dr. Elleh's webinar from July 12, 2022, available in the WGU Courseware Resources, helped me understand how to use SelectKBest and decide on the appropriate 'k' value.

For finding the best 'k' value for the KNN classification, I referred to the Hyperparameter Tuning section in Hugo Bowne-Anderson's DataCamp class material.

To compute the AUC score for my KNN model, I used the explanation and code from Hugo Bowne-Anderson's DataCamp class on AUC Computation.

Lastly, I used the code from Hugo Bowne-Anderson's DataCamp class on Plotting an ROC Curve to visualize the AUC score and the performance of the KNN model.

[Scikit learn documentation for KNeighborsClassifier](https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html) was used for research into the weights and algorithms underpinning KNN classification.