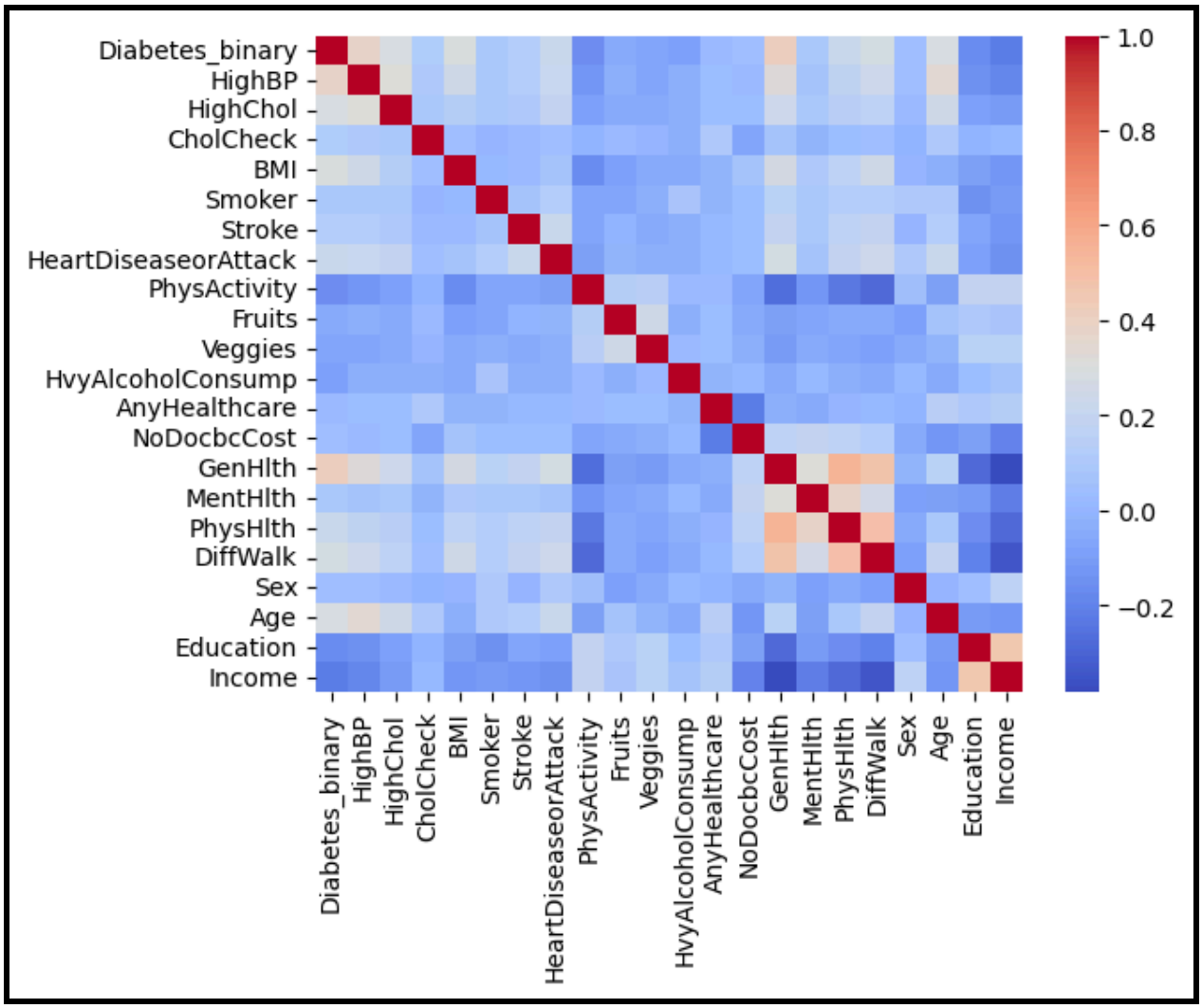
Main objective of the analysis:

- The main objective of my analysis will be concerning diabetes. It will be a prediction focused on determining if someone is pre-diabetic based on attributes.
- People who will benefit from this are those individuals who are pre-diabetic. People who are pre- Type 2 diabetes who know their condition beforehand can take measures to prevent this from occurring, which will save them medical expenses in the long run.
- Also, Insurance companies can also benefit because when their customers get diabetes, they will have to cover some of their expenses. Of course, if their customer never gets diabetes, they won't have to be paying these expenses.
- The analysis will contain 3 different neural networks
    - Multilayer Perceptron Neural Network
    - Convolutional Neural Network
    - Convolutional Neural Network with Residual Net

Brief description of the data set:

- Dataset was obtained from kaggle and rated 10/10 usability. https://www.kaggle.com/datasets/alexteboul/diabetes-health-indicators-dataset?resource=download
- The CSV I used has just over 76,000 rows of data. I used the 50-50 class split dataset which downsampled from the original dataset, where there were a lot more non-diabetics than pre-diabetics which really messed up my model.
- In this analysis, I will be using all 21 features in the dataset(BMI, Smoking, Blood Pressure, etc.) in order to train three different neural networks, each with the metric of **the recall function**. I chose this error metric because we really want to punish False-Negatives, or cases where we misidentify a pre-diabetic by saying they are not.
- To gain an initial understanding ofthe correlation within the data, I used the seaborn correlation heatmap in order to view the correlation coefficients of each of the features with respect to all other features:



From this, we see that the Gen Hlth column is correlated to some extent with the binary diabetes column.

Brief summary of data exploration and actions taken for data cleaning and feature engineering:

- Some columns are continuous, whereas some are binary. To solve this, I used the standardScaler library to scale the models for logistic regression and Support Vector Machines
- The feature of interest, the 'Diabetes' column,actually has 3 values. 0 (non-diabetic), 1(pre-diabetic), and 2(diabetic). Because the analysis is focused on predicting pre-diabetes only, I dropped rows identifying already diabetic individuals. Then, as stated earlier, I downsampled so that there were an equal number of these two classes.

Summary of training at least three different classifier models:

1. **Multilayer Perceptron Neural Network**
    - All layers use sigmoid activation function
    - **3 Hidden Layers**
    - 1st Hidden Dense Layer: 64 neurons
    - 2nd Hidden Dense Layer: 32 neurons
    - 3rd Hidden Dense Layer: 64 neurons
    - Dropout Layer (.3)
    - Output layer: 1 neuron
    - Adam optimizer utilized
    - Binary Crossentropy loss function utilized
    - **10 Epochs(Iterated through data 10 times)**
    - Batch size = 32 samples
    - Final classification report:

	precision	recall	f1-score	support
0.0	0.8005	0.6714	0.7303	7090
1.0	0.7156	0.8317	0.7693	7049
accuracy			0.7513	14139
macro avg	0.7581	0.7516	0.7498	14139
weighted avg	0.7582	0.7513	0.7497	14139

2. **Convolutional Neural Network**

    - **All layers use sigmoid activation function**
    - First Conv1D Layer: 64 filters, kernel size = 3
    - Second Conv1D Layer: 32 filters, kernel size = 3
    - First MaxPooling1D: Pool size = 2.
    - Second MaxPooling1D: Pool size = 2.
    - 1 Hidden Dense Layer
    - 1st Hidden Dense Layer: 64 neurons
    - Output layer: 1 neuron
    - Adam optimizer utilized
    - Binary Crossentropy loss function utilized
    - **10 Epochs(Iterated through data 10 times)**

- Batch size = 32 samples
- Final classification report:

	precision	recall	f1-score	support
0.0	0.7925	0.6533	0.7162	7090
1.0	0.7036	0.8279	0.7607	7049
accuracy			0.7404	14139
macro avg	0.7481	0.7406	0.7385	14139
weighted avg	0.7482	0.7404	0.7384	14139

3. Convolutional Neural Network with ResNet

- **All layers use sigmoid activation function**
- **Essentially the same as #2, but with two residual blocks**
- Residual block 1: 32 filters, kernel size = 3
- Residual block 2: 64 filters, kernel size = 3
- Final classification report:

	precision	recall	f1-score	support
0.0	0.8119	0.6539	0.7244	7090
1.0	0.7089	0.8476	0.7721	7049
accuracy			0.7505	14139
macro avg	0.7604	0.7508	0.7482	14139
weighted avg	0.7605	0.7505	0.7482	14139

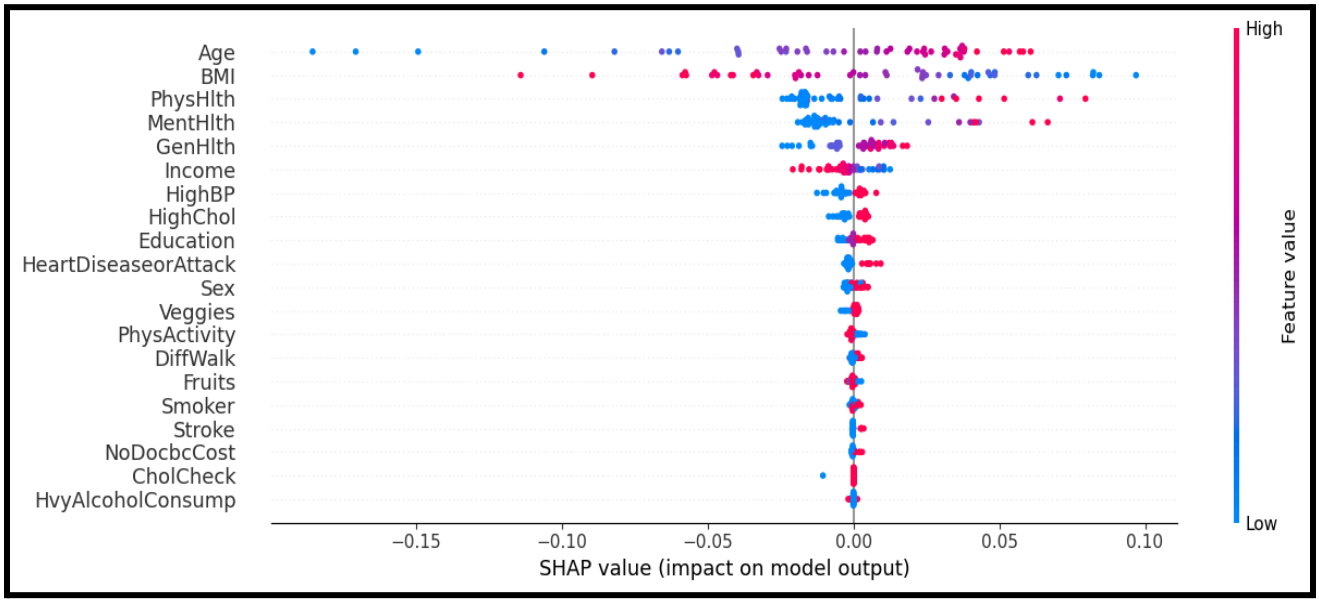
Final Recommendation:

Out of all the models, **I recommend the basic Multilayer Perceptron Neural Network.** This is because it has a higher recall score then the convolutional neural network, and a close enough recall score to the CNN with Resnet to where its simplicity makes it key. Additionally, the dropout layer ensures to some degree that the model did not overfit to the data while training. This means that it is very likely that this model will generalize well to new data and not just be good at the existing data.

Summary Key Findings and Insights:

Looking at the MLP, we can use black box interpreter models to determine an estimate for the importances of different features. I used a SHAPley interpreter, which sends in multiple data points and measures the change in

predictions in order to identify the features which contribute most to changing the outcome. The output of the SHAP is shown below:



Suggestions for next steps in analyzing this data:

The models I used are powerful, but still more might be done in order to overall improve each of their performances.

- A powerful scoring metric I could have used would be the ROC curve. This would allow for one to find the best performing model in terms of overfitting error and complexity by choosing a specific point on the curve.
- I also could have tried different convolutional Neural Network structures for Models #2 & 3. Choosing a different structure such as the AlexNet may or may not have produced better results. I only tried the Residual Net for this analysis.
- A third area of possible improvement would be switching the sigmoid activation functions with other known functions, such as ReLu or Hyperbolic tangent. This might be able to give the model more variety rather than just sticking with only sigmoids.
- The last possible area of improvement that came to my mind would be trying different cost functions or scoring metrics. I only used recall, but maybe a different would would actually lead to better overall reports.

==========