SNOWBALL
Organized by DATA Club

CDC Diabetes Prediction Based on Health Indicators

Luke Abbatessa & Rajendra Goparaju

Table of Contents

Background/How the Idea Came to Be

- We were initially interested in a science-based, data science-focused project
- After some discussion, we agreed upon a project focused on healthcare
- After researching publicly accessible datasets in the UC Irvine Machine Learning Repository, we stumbled upon a dataset titled “CDC Diabetes Health Indicators”
- We chose this dataset in particular because

  -The dataset is only eight years old

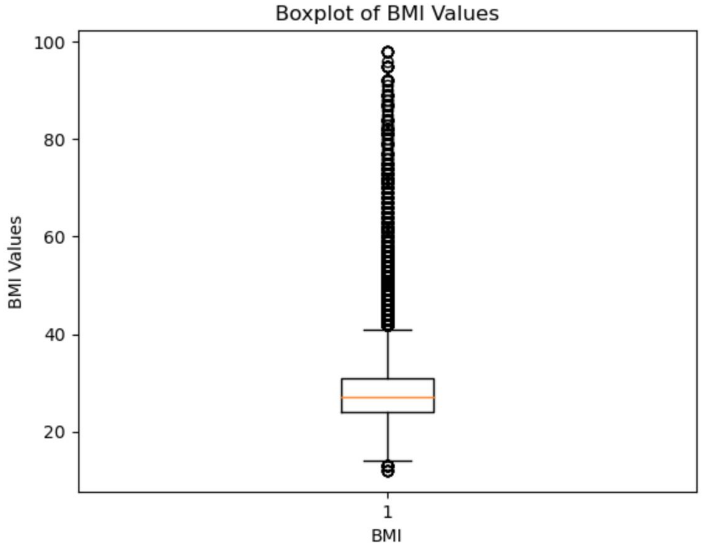
  -It consists of 253,680 instances (more data = stronger ML/DL models), each of which represents a person participating in a study (real data)

  -Its creation was funded by the CDC, and it has no missing values (trustworthy)

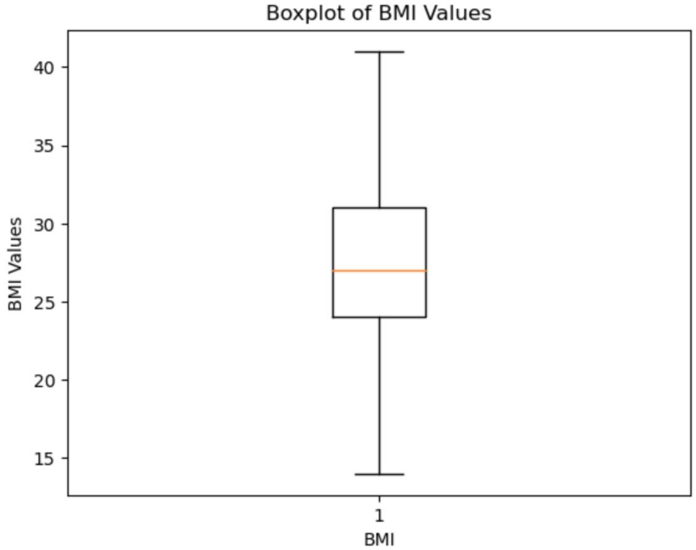
Methodology - Data Cleaning/Preprocessing

In an effort to clean the data, we:

- Changed the variable data types from float-based to integer-based
- Removed outliers from the variable "BMI" using the Interquartile Range (IQR) Method
- Dropped duplicate rows from the Pandas DataFrame
- Shuffled the data to eliminate the possibility of any biases, such as when calculating accuracy during model implementation
- Standardized the non-binary features and dropped the non-standardized features from the dataframe



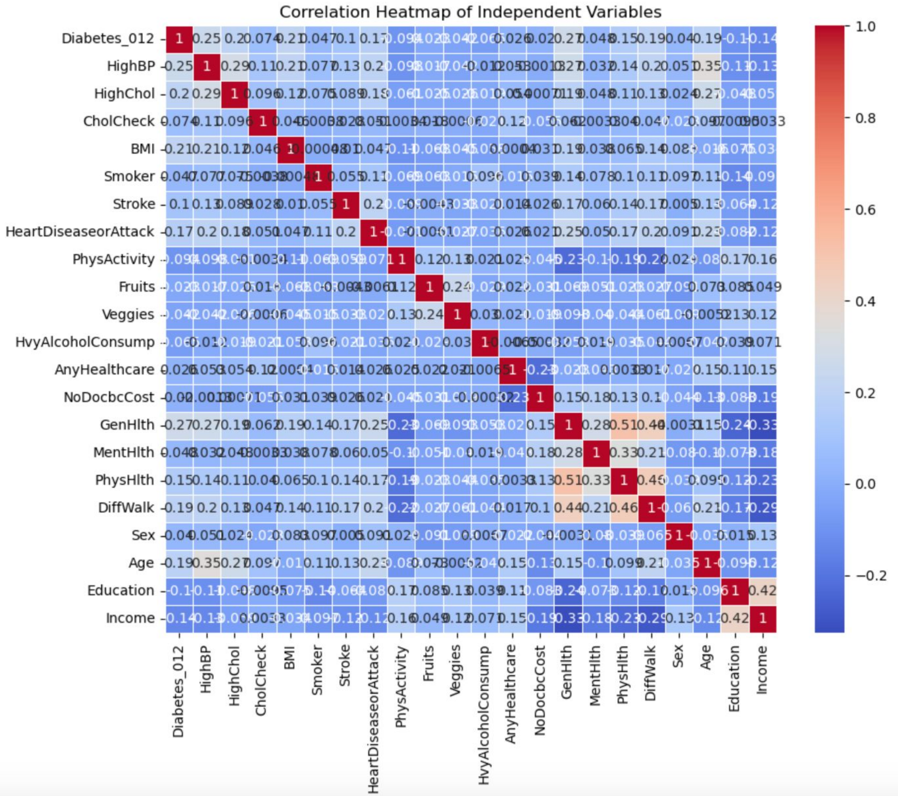
Before removing outliers from "BMI"



After removing outliers from "BMI"

Methodology - Exploratory Data Analysis

- We plotted a correlation heatmap to determine the correlations between the features present in the dataset



- Since the highest correlation coefficient between two independent variables was 0.51 (considerably weak positive correlation), no independent variables were confounded, thus we kept all features for ML/DL model development

Methodology - Data Division/Model Selection

Data Division

- Once we isolated the features (X) from the target (y), we performed a train-test split on the data, allocating 70% of the data for training and the remaining 30% for testing
- We followed up the train-test split by performing another split on the testing data, allocating 50% of the data for validation and the remaining 50% for testing
- For each of the ten different models we implemented, we predicted the target variable both for the cross-validation dataset as a baseline and for the test dataset as a final result
- This data division was repeated over two iterations (once when working with all the features, a second time after feature selection was implemented)

Model Selection

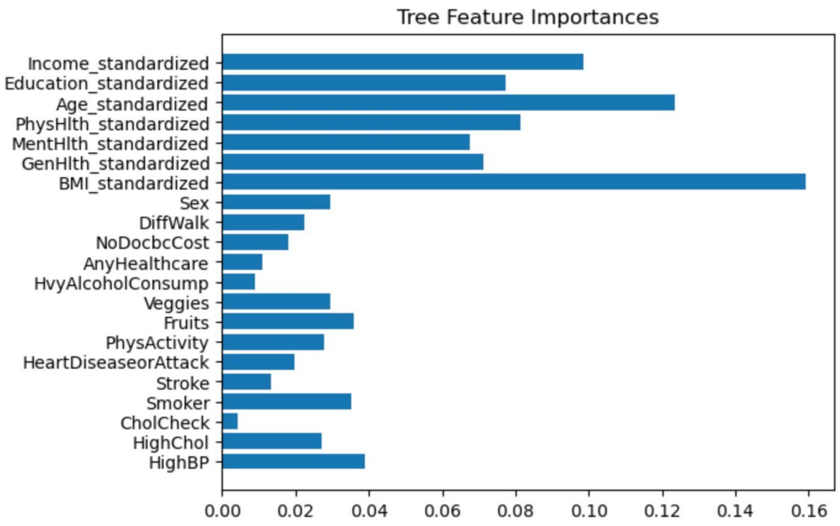
- In the beginning, we chose SciKit-Learn's Decision Tree Classifier, SciKit-Learn's Random Forest Classifier, Keras' Neural Networks Classifier, and the XGBoost Classifier because we were familiar with all four of these models prior and knew they were all capable of multi-class classification
- As the project went on, we decided to forego the Decision Tree Classifier because the Random Forest Classifier produced better metrics

Methodology - Feature Selection

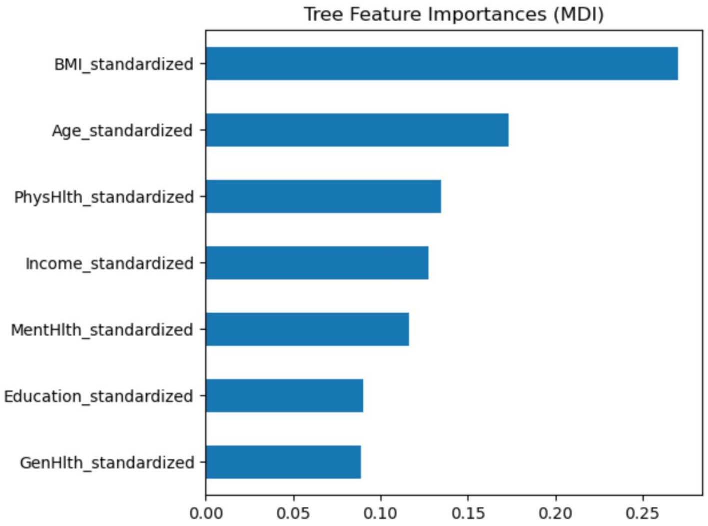
- We implemented two techniques, both with the use of SciKit-Learn’s ExtraTreesClassifier:

1. Tree-based feature selection
2. Feature selection as part of a pipeline/tree’s feature importance from Mean Decrease in Impurity (MDI)

- Both techniques informed us of the same most important features: ‘BMI_standardized’, ‘GenHlth_standardized’, ‘MentHlth_standardized’, ‘PhysHlth_standardized’, ‘Age_standardized’, ‘Education_standardized’, and ‘Income_standardized’



Tree-based feature selection



Feature selection as part of a pipeline/tree’s feature importance from Mean Decrease in Impurity (MDI)

Methodology - Hyperparameter Tuning

- We used GridSearchCV to perform hyperparameter tuning for the Random Forest Classifier, Neural Networks Classifier, and the XGBoost Classifier

Random Forest Classifier

- “n_estimators” (The number of trees in the forest): 50
- “criterion” (The function used to measure the quality of a split): “gini”
- “max_depth” (The maximum depth of the individual trees in the forest): 10
- “min_samples_split” (The minimum number of samples required to split an internal node): 5

Neural Networks Classifier

- “units” (The number of neurons in a particular layer): 32
- “learning_rate” (Controls the size of the steps taken during the optimization process): 0.001
- “batch_size” (The number of training examples used in one iteration): 32
- “epochs” (Number of times the learning algorithm will work through the entire training dataset): 10
- “optimizer” (An algorithm that adjusts the model’s weights during training to minimize the loss function): “adam”
- “activation” (A function that allows the neural network to learn complex patterns): “sigmoid”

XGBoost Classifier

- “learning_rate”: 0.1
- “n_estimators”: 100
- “max_depth”: 5
- “min_child_weight” (Minimum sum of instance weight needed in a child): 1
- “subsample” (Fraction of samples used for fitting the individual base learners): 1.0
- “colsample_bytree” (Fraction of features used for fitting the individual base learners): 1.0
- “gamma” (Minimum loss reduction required to make a further partition on a leaf node): 0.1

Results

SciKit-Learn's Decision Tree Classifier (all features)

```
Below are the metric scores for the test dataset:
{'accuracy': 0.7442635870388894, 'sensitivity': 0.7442635870388894, 'specificity': 0, 'precision': 0.75981352472356
33, 'f1-score': 0.7518045629032043}
```

Results (cont.d)

SciKit-Learn’s Random Forest Classifier (all features)

```
Below are the metric scores for the test dataset:
{'accuracy': 0.8283168136764573, 'sensitivity': 0.8283168136764573, 'specificity': 0, 'precision': 0.77334365746626
35, 'f1-score': 0.7891639918336054}
```

Results (cont.d)

Keras' Neural Networks Classifier (all features)

```
Below are the metric scores for the test dataset:
{'accuracy': 0.8398047952471887, 'sensitivity': 0.8398047952471887, 'specificity': 0, 'precision': 0.78881350764842
73, 'f1-score': 0.7866034144670544}
```

Results (cont.d)

XGBoost Classifier (all features)

```
Below are the metric scores for the test dataset:
{'accuracy': 0.8388348337425359, 'sensitivity': 0.8388348337425359, 'specificity': 0, 'precision': 0.78798324407653
42, 'f1-score': 0.7961112450746823}
```

Results (cont.d)

SciKit-Learn’s Random Forest Classifier (selected features)

```
Below are the metric scores for the test dataset:
{'accuracy': 0.8169803885908278, 'sensitivity': 0.8169803885908278, 'specificity': 0, 'precision': 0.7663573390499506, 'f1-score': 0.7842671351281505}
```

Results (cont.d)

Keras' Neural Networks Classifier (selected features)

```
Below are the metric scores for the test dataset:
{'accuracy': 0.8373192688915159, 'sensitivity': 0.8373192688915159, 'specificity': 0, 'precision': 0.77917386584879
63, 'f1-score': 0.7787016893781806}
```

Results (cont.d)

XGBoost Classifier (selected features)

```
Below are the metric scores for the test dataset:
{'accuracy': 0.8361068170106999, 'sensitivity': 0.8361068170106999, 'specificity': 0, 'precision': 0.7763656813742675, 'f1-score': 0.7826595597838534}
```

Results (cont.d)

SciKit-Learn’s Random Forest Classifier (all features and optimal hyperparameters)

```
Below are the metric scores for the test dataset:
{'accuracy': 0.8390773241186991, 'sensitivity': 0.8390773241186991, 'specificity': 0, 'precision': 0.78736816568644
91, 'f1-score': 0.7807881132263675}
```

Results (cont.d)

Keras' Neural Networks Classifier (all features and optimal hyperparameters)

```
Below are the metric scores for the test dataset:
{'accuracy': 0.8398047952471887, 'sensitivity': 0.8398047952471887, 'specificity': 0, 'precision': 0.79304553574094
09, 'f1-score': 0.8041445463036988}
```

Results (cont.d)

XGBoost Classifier (all features and optimal hyperparameters)

```
Below are the metric scores for the test dataset:
{'accuracy': 0.839683550059107, 'sensitivity': 0.839683550059107, 'specificity': 0, 'precision': 0.788729235209418
3, 'f1-score': 0.7941229161744714}
```

Results (cont.d)

Model/Metric	Accuracy	Sensitivity	Specificity	Precision	F1-Score
SciKit-Learn's Decision Tree Classifier (all features)	0.74426358 70388894	0.744263587 0388894	0	0.75981352 47235633	0.7518045 629032043
SciKit-Learn's Random Forest Classifier (all features)	0.82831681 36764573	0.828316813 6764573	0	0.77334365 74662635	0.7891639 918336054
Keras' Neural Networks Classifier (all features)	0.83980479 52471887	0.839804795 2471887	0	0.78881350 76484273	0.7866034 144670544
XGBoost Classifier (all features)	0.83883483 37425359	0.838834833 7425359	0	0.78798324 40765342	0.7961112 450746823
SciKit-Learn's Random Forest Classifier (selected features)	0.81698038 85908278	0.816980388 5908278	0	0.76635733 90499506	0.7842671 351281505
Keras' Neural Networks Classifier (selected features)	0.83731926 88915159	0.837319268 8915159	0	0.77917386 58487963	0.7787016 893781806
XGBoost Classifier (selected features)	0.83610681 70106999	0.836106817 0106999	0	0.77636568 13742675	0.7826595 597838534
SciKit-Learn's Random Forest Classifier (all features and optimal hyperparameters)	0.83907732 41186991	0.839077324 1186991	0	0.78736816 56864491	0.7807881 132263675
Keras' Neural Networks Classifier (all features and optimal hyperparameters)	0.83980479 52471887	0.839804795 2471887	0	0.79304553 57409409	0.8041445 463036988
XGBoost Classifier (all features and optimal hyperparameters)	0.83968355 0059107	0.839683550 059107	0	0.78872923 52094183	0.7941229 161744714

Keras' Neural Networks Classifier (all features and optimal hyperparameters) proved to be the best model at predicting whether someone was diabetic, pre-diabetic, or healthy based on health indicators

Future Work

If we were to do anything differently, we would:

- Consider using a graphics processing unit (GPU) to perform more in-depth hyperparameter tuning and/or train the models quicker
- Consider incorporating self-developed models to compare against SciKit-Learn and Keras' counterparts, in addition to XGBoost
- Consider incorporating our own hyperparameter tuning to compare against GridSearchCV