Diabetes Prediction Report

Introduction & Background

The main goal of our project is to create a web application that can predict if a person has Type 2 diabetes or not based on specific characteristics. Two main things motivated this project. For one, we wanted to understand how factors such as a patient's genetic predispositions and lifestyle choices could give healthcare providers insights into whether the patient has diabetes. Secondly, we wanted to create an application that could be easily used by the average person and could help them decide if they should visit a healthcare provider to confirm their symptoms. Therefore, when deciding on features to use, we had to make sure they were predictive of whether someone had diabetes and that the features were things that the user could easily measure themselves and provide as inputs to our application.

To tackle this, we found a dataset on Kaggle called the Diabetes Health Indicators Dataset (linked in the References section at the end of the report). We decided to use the 'diabetes_012_health_indicators_BRFSS2015' CSV file, which contains the data for our project.

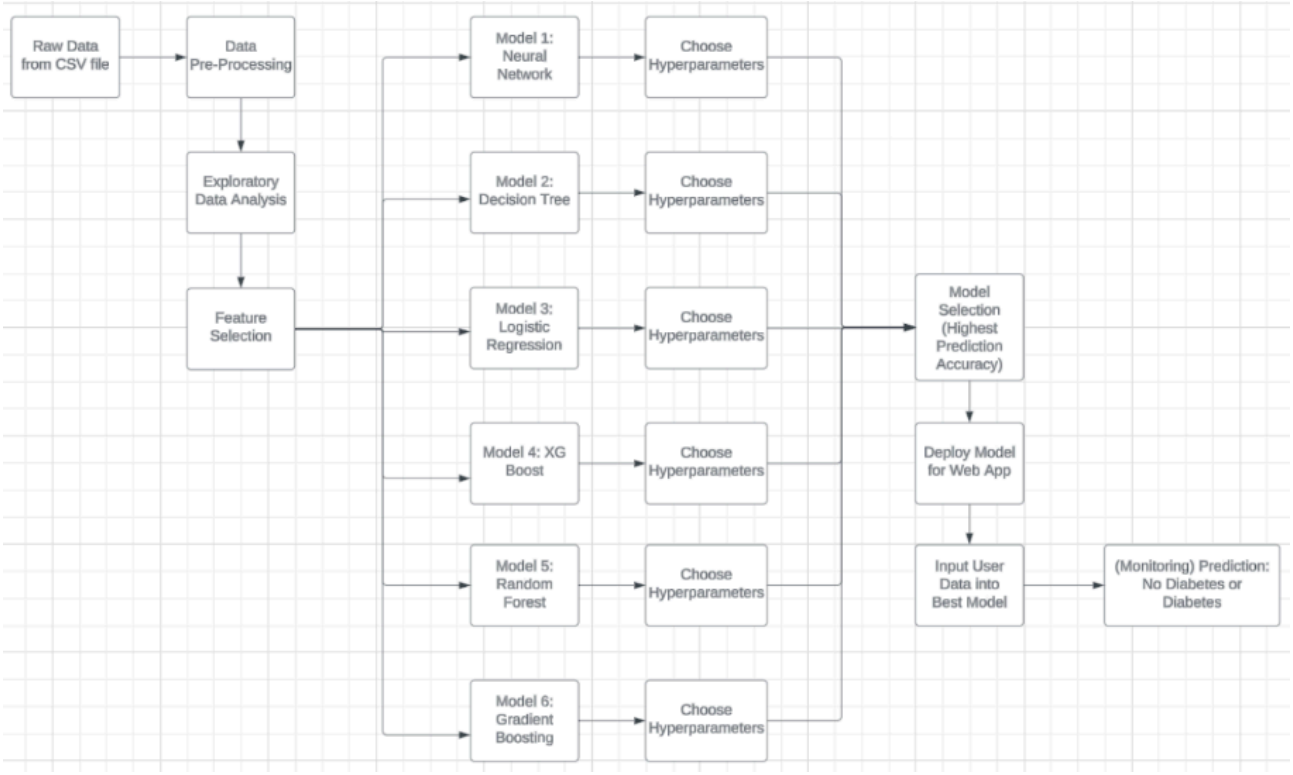
We discovered that our dataset consisted of 253680 observations – 21 features and 1 outcome variable. This outcome variable was categorical, with 3 possible groups: no diabetes, prediabetes, and diabetes. However, this dataset was imbalanced, meaning there were different amounts of observations in each outcome group, so we resorted to downsampling. We also decided that we only wanted to determine if a person had no diabetes or diabetes, a binary classification problem since it was more useful for our application, and the prediabetes happened to have far fewer observations.

Here's a brief overview of how we define the models we'll be using. Neural networks consist of multiple layers of perceptrons that allow for nonlinear relationships and learn weights and thresholds during training. Logistic regression assigns an observation to one group based on which one it has the highest probability of belonging. Decision trees keep splitting on features that give the most information gain, while random forest is a bagging ensemble technique that combines multiple decision trees. Instead of bagging, gradient boosting is a boosting ensemble technique, and XGBoost is a more efficient and performance-optimizing version.

Methodology

Before performing EDA, we had a few preliminary steps to take as part of pre-processing the data. Since our focus was on building models that predicted whether the user had diabetes or not, we decided not to include the "Prediabetes" data from our dataset. Prediabetes was also the category that had the least count of people recorded, so it made sense for it to be excluded. Additionally, the dataset was balanced in order to have an equal number of observations for both "Diabetes" and "No Diabetes" (35,097 observations each). We did not have to clean the data further since the description on Kaggle mentioned that the dataset was already cleaned, and there appeared to be no NA/missing values. Figure 1 below provides an outline of the entire process from completing EDA and building the models to the deployment and construction of a local host web application:

Figure 1: System Implementation



EDA:

For EDA, our team mainly focused on creating visualizations for the data to narrow our focus down to which main features served as good predictors of diabetes. We created multiple

count plots, pair plots, histograms, and violin plots to understand the interactions between certain features. Our domain knowledge related to diabetes and its causative factors also allowed us to choose our main predictors.

From the count plot for high blood pressure (Figure 2), it can be inferred that there is a noticeable pattern between diabetes and BP since the highest count of BP also correlated with those who recorded they have diabetes. Moreover, we knew that blood pressure and diabetes were associated with insulin resistance, strengthening our assumption that this could be a strong predictor. High cholesterol provided a visualization fairly similar to that of blood pressure, where there was a clear increase in the number of people who reported they have high cholesterol and diabetes– cholesterol is also associated with insulin resistance. This can be seen in Figure 3 below:

Figure 2: High BP Count Plot

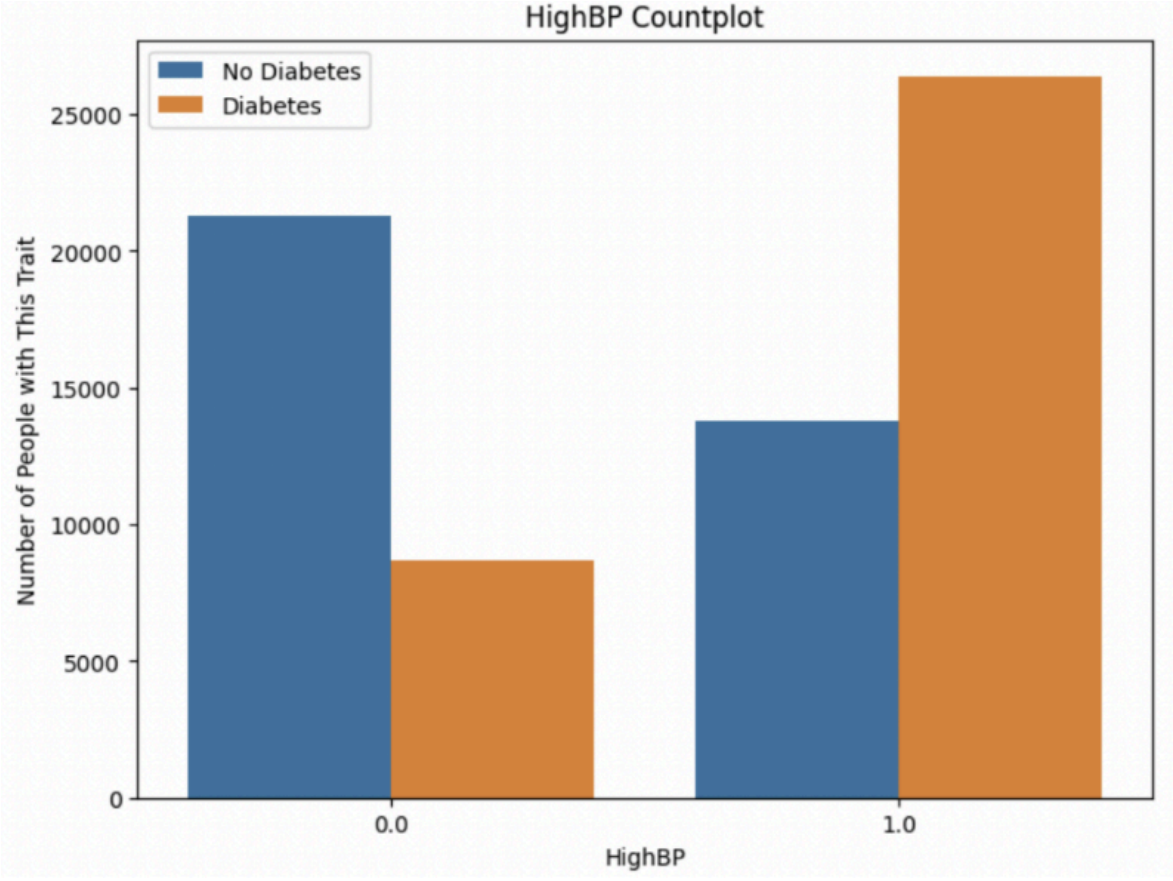
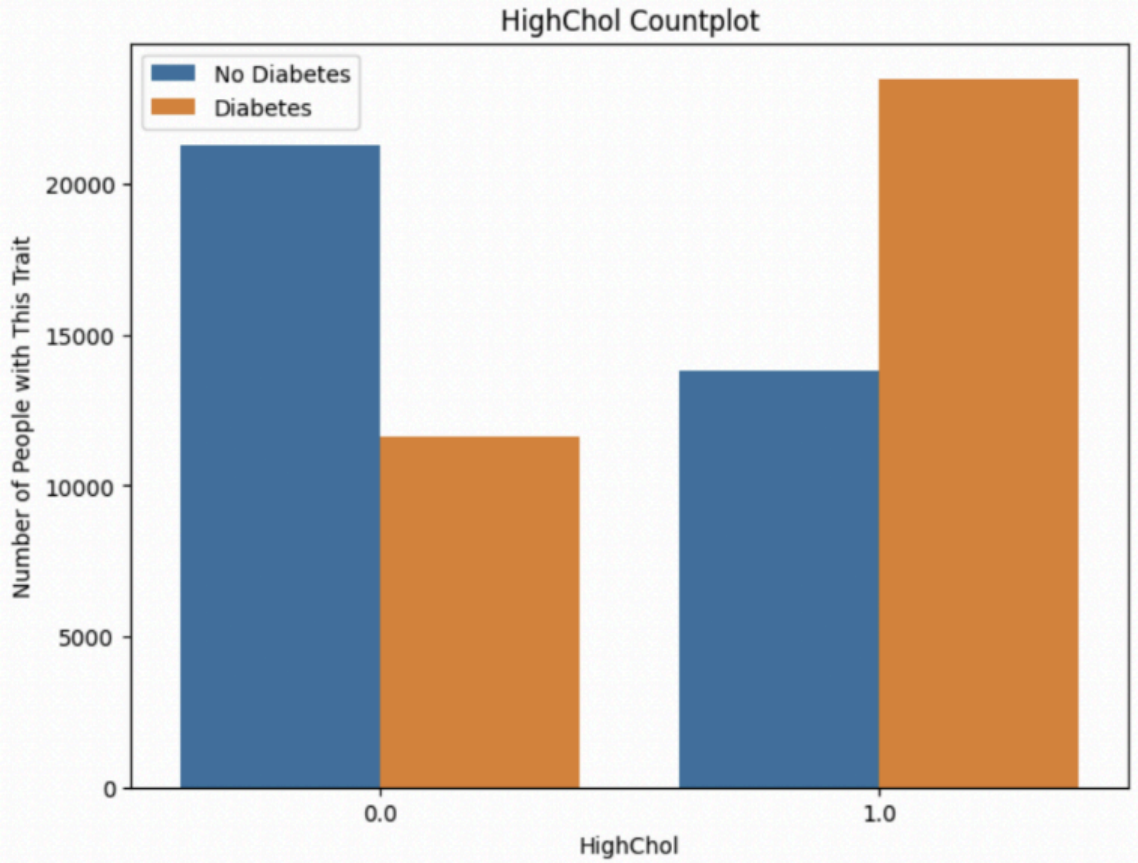


Figure 3: High Cholesterol Count Plot



Albeit a slight skew to the left, the count plot for age (Figure 4) displayed a bell curve, suggesting a normal distribution. Based on our prior knowledge about patterns relating age to diabetes, middle-aged and older individuals are more susceptible to diabetes, making it a predictor that we could potentially focus on, but not as strong of an indicator as blood pressure or cholesterol. The sex count plot (Figure 5) displayed that women were more susceptible to diabetes, and we knew from prior understanding that women could have diabetes at times of pregnancy or menopause due to hormonal changes. Although not a direct cause of diabetes, we believed it would be a risk factor due to biological differences.

Figure 4: Age Count Plot

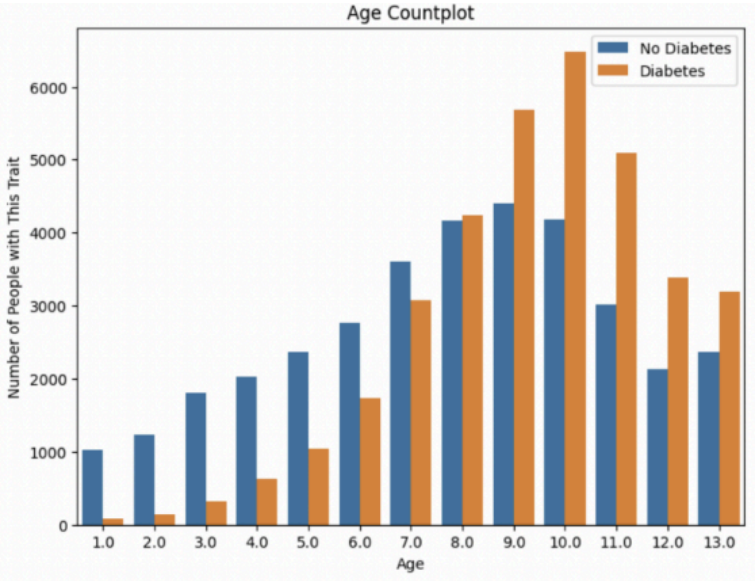
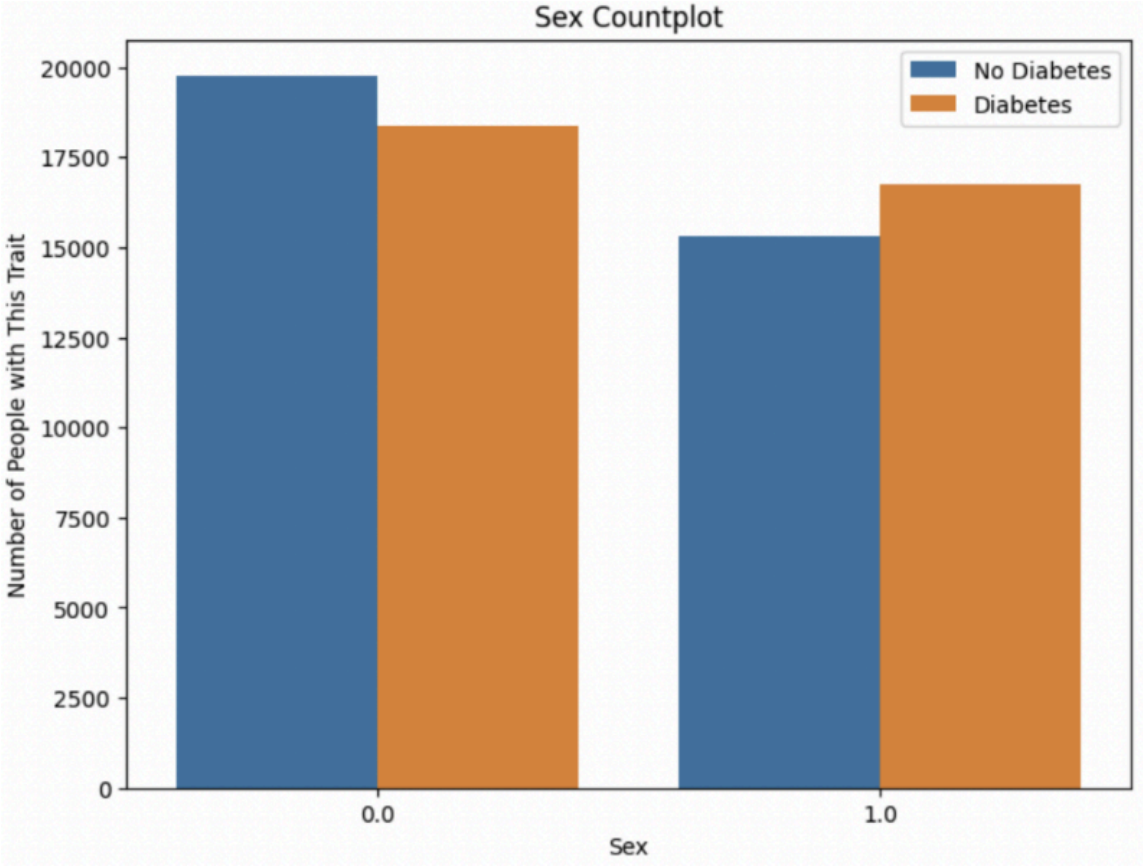


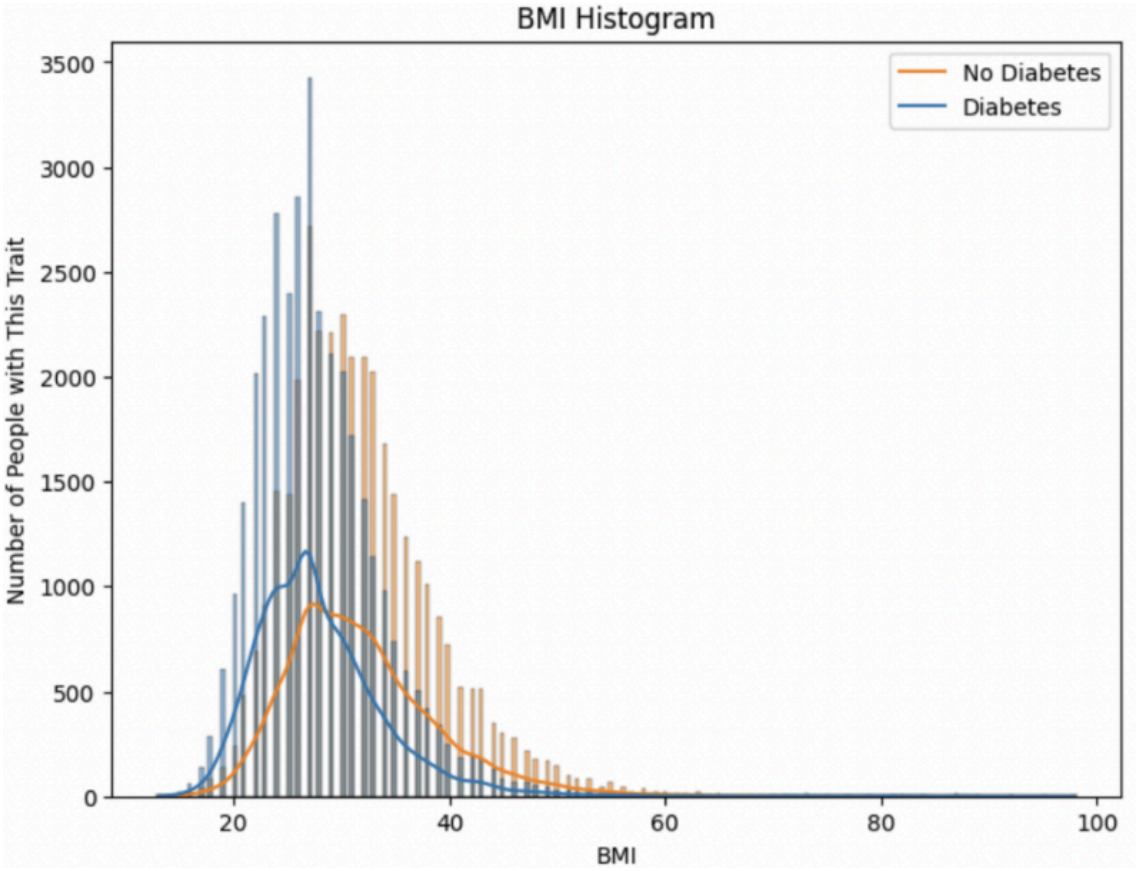
Figure 5: Sex Count Plot



The BMI histogram (Figure 6) was also very informative. The bell curve distribution suggested that most individuals fell within a specific BMI range where the risk of developing

diabetes was the highest. We created a similar plot to that of BMI to understand more about any patterns in physical health and mental health recorded by patients, but were unsuccessful in noticing any significant trend since the mental health, for example, displayed that the most number of people recorded a poor mental health status, regardless of whether they have diabetes or not.

Figure 6: BMI Histogram



Model Building & Selection:

Before we began building our models, we decided to select the main features we wanted to incorporate in our models. Based on the EDA, the important features we wanted to select were high cholesterol, high BP, and BMI. However, we wanted to confirm those choices with a decision tree model which would give us the most important features. Based on the decision tree, the top three features were BMI, highBP, and age. This confirmed our idea about which predictors to use. Since our final goal was to create a local host that would take user input and predict the presence of diabetes, we also wanted to include demographic predictors like Age and

Sex. In total, our five predictors that we chose to work with were BMI, high BP, high cholesterol, Age, and Sex.

We chose six machine learning models to work with: logistic regression, decision trees, neural network, random forest, gradient boosting, and XGBoost. We chose these models mainly to capture more nuances in the data. The logistic regression model served as a simple starter model that provided a baseline for comparison. The decision tree model was selected for its interpretability and automatic feature selection capabilities. The neural network model was chosen for its ability to capture complex relationships that might exist within our dataset. The ensemble methods (random forest, gradient boosting, and XGBoost) were selected for their robustness, especially in handling complex datasets and capturing interactions between features, as well as using voting mechanisms to improve accuracy. These classification models all achieved an accuracy of around 70%. Unsurprisingly, the top 3 models that achieved the highest accuracies were the ensemble methods: XGBoost, gradient boosting, and random forest, in order, starting with highest accuracy.

Although XGBoost seemed like our most promising choice due to the model having the highest accuracy, we also wanted to confirm the feature importance behind the three models. Because our data predicts Type 2 Diabetes, we knew that high blood pressure, high cholesterol, and age were the strongest determiners of Type 2 Diabetes. By calling for the feature importance on these models, the model with the three top features we were looking for was XGBoost. Since XGBoost had the highest accuracy and had the highest dependencies on the features we knew were important, we chose XGBoost as our final model to work with while implementing the local host.

Local Host Launch:

We decided on launching our user interface through a local host rather than creating an independent web application. Our user interface and local host went through multiple phases of building and testing. With HTML, we first created the boxes that would contain all of the predictors and their respective input boxes and bubbles. After creating our base, we moved onto finding a way to incorporate our XGBoost predictive model. To do this, we imported our HTML code as a template in Jupyter Notebook and used Flask to allow the template to manifest as a

local host web page. Then, we used the pickle module which allowed us to load the model into the local host and use user input as new values to feed into the model. Programming the buttons to submit the input values and return a certain image based on the output of the XGBoost model required a lot of trial and error. Once we were satisfied with the results, we decided to work on the more intricate details of our user interface. Our goal was to allow the user to run as many numbers and tests as they want. However, at that point, users were only able to run their numbers once, having to reload the local host to test new values. We found a way to wipe the data once submitted by moving all of our input boxes and bubbles under a form element, allowing for the model to accept new numbers for a new prediction. Finally, to make the user interface more visually appealing, we added graphics and also adjusted the borders on our boxes to resemble clipboards to go along with the medical theme.

Results

Despite the choices we had to make around the limitations introduced by our data, we were satisfied with the outcomes of our project. Since we had chosen XG Boost as our main model, we were able to gain insights into the importance of features, which helped us understand the underlying factors. The key features of XGBoost were age, high cholesterol, and high blood pressure. While this aligns with what we had expected, age seemed to impact this model the most; this makes sense because age has a strong influence on Type 2 Diabetes. We successfully

incorporated these features as part of our web application to get user input, allowing the model to cater to each individual's predictions. In general, the robustness of the model was also good as it yielded expected results for the myriad of inputs we provided to test the web application. Hyperparameter tuning by manual trial and error methods was incorporated in order to mitigate over and underfitting. Although our web application provided fairly accurate results, the frequency of accurate predictions could have been improved to meet our expectations since the project's goal was to enable users to realize whether they required additional and urgent medical care. Overall, our model does yield the desired results to an extent. Still, we suggest against entrusting it with possible health perils, as it is always best to consult a healthcare provider before confirming whether one has diabetes. Our predictive model is merely a tool to identify potential risks of disease susceptibility.

Discussion & Conclusion

After turning the 'diabetes_012_health_indicators_BRFSS2015' dataset into a balanced one, selecting features, and building 6 binary classification machine learning models (Neural Network, Decision Tree, Random Forest, Logistic Regression, Gradient Boosting, and XGBoost), we settled on the model that gave us the highest accuracy in predicting Type 2 diabetes. This turned out to be the XGBoost model, reaching a 71.38% prediction accuracy, and we used it as the model to feed into our web application. Since our original data was balanced, there was a 50% chance for a model to guess that a person belonged to the diabetes group. However, our tuned model's results are significant because we've raised this chance of predicting the right group to 71.38%, which is much higher than 50%. Consequently, we argue that a person looking at the results produced by our application, can be more certain that our prediction is accurate, compared to a random guess.

From this project, we also learned after fitting and tuning the 6 models, that the accuracies all remained around 70%, so it didn't seem like we could improve on that number. This limitation could be due to the fact that we initially chose an imbalanced dataset and had to randomly sample from the no diabetes group to fix this dataset imbalance. Some other important choices we made were in reducing the number of features after reading the documentation and considering the web application, as well as in choosing hyperparameters by manual trial and

error. We learned how to create a local host web page that was not only functional but also visually appealing. Moreover, we overcame the challenge of having to incorporate both HTML and Python to allow for our model to be interactive. The process of creating our local host required a lot of time and research.

Perhaps with more time, we could try improving our models by oversampling instead of downsampling to see if the accuracy would improve. We could also try combining this dataset with other related datasets online so we get more data, and/or try using Randomized Search or Grid Search to pick the best hyperparameters.

References:

Dataset:
https://www.kaggle.com/datasets/alexteboul/diabetes-health-indicators-dataset?resource=download&select=diabetes_012_health_indicators_BRFSS2015.csv