Diabetes Health Indicators

Pratyush Jaishanker Will Dolan

Abstract

Diabetes is a condition that threatens the health of many in the United States and around the world. Our task was to build a machine learning model to quickly and accurately alert patients and doctors of the possibility of diabetes or pre-diabetes in a patient. We did this by implementing four different machine learning methods: Logistic Regression, Decision Tree, Neural Network, and XGBoost Models. While all four methods had comparable accuracy, the XGBoost model was far and away the best in terms of recall, a measure of the rate of detection of true positives, which is crucial in our task of predicting diabetes. The strength of XGBoost for this task has been corroborated by research by others on similar datasets, and future research into this problem should look at using similar models while incorporating more focused feature selection to improve performance.

1. Introduction

Diabetes is a condition that causes the body to alter its production of insulin. This key hormone regulates energy use in cells by controlling the amount of glucose, a sugar, in the blood. There are many types of diabetes, including type I, type II, and gestational. Additionally, prediabetes is a condition in which blood sugar is elevated but not enough for a diagnosis of type II diabetes. The CDC estimates that 1 in 3 Americans have some level of prediabetes, while 8 in 10 of those do not know they have it. Prediabetes increases the risk of more severe conditions, including type II diabetes, heart attack, and stroke (Centers for Disease Control and Prevention, 2023b).

Our machine-learning model will aim to alert doctors that a patient has an elevated probability of having prediabetes or diabetes so the doctors can prescribe preventative measures to manage or reverse the condition. The model will input the patient's demographics (including age, medical history, etc.) and output whether or not the model thinks they have a form of diabetes (or prediabetes), making this a binary classification task. The model could also output its confidence in its answer as an additional check.

Our progress includes implementing four methods: Logistic Regression, Decision Tree, Neural Network, and XGBoost models. All implementations returned machine learning models for the diagnosis of diabetes/prediabetes in patients based on the information listed in the previous paragraph. These implementations yield a decent accuracy, but as we will describe in 3, our desired metric is recall, as accuracy fails to be a good metric due to class imbalance. Using this metric, the latter model and Decision Tree performed the best. Our detailed description of the work we have done and the process by which we have achieved it will be outlined within the rest of this paper.

2. Related Work

Many studies have looked at the question of predicting diabetes using patient data. One study by I. Tasin (2022) compared many machine-learning techniques on a dataset from the Pima Indian population and an additional sample from Bangladesh. Some models they used include SVM, Random Forest, Logistic Regression, and KNN. They found that the XGBoost architecture gave the best accuracy at 81% and an F-1 score of 0.81. However, because they trained on very specific populations, their conclusions may not hold when applied to the general populace. Additionally, they did not use any deep learning techniques, an area we can expand upon.

Another study by M. K. Hasan (2020) also analyzed the Pima Indian dataset, used similar methods, and incorporated deep learning. Their best model was an ensemble model combining the AdaBoost and XGBoost architectures with an AUC of 0.950. Additionally, this paper conducted a hyperparameter grid search to identify the best-performing parameters. Both ensembling and hyperparameter search are methods we can apply to our model. Because the Pima Indian dataset has $<$1000 examples, the high metrics may also result from a lack of data.

Our research aims to evaluate 4 methods used by these papers (Logistic Regression, Decision Tree, Neural Network, and XGBoost) on this related but separate dataset to determine the robustness of these methods in the task of diabetes classification.

Through our work and our final model, we have achieved near-parity with the previous literature. Further discussed in our results, we had a weighted f1-score of 0.78 as compared

to the 0.81 given in (I. Tasin, 2022). Our recall was 0.75 compared to their 0.79. One thing to note is that our datasets were different, and so our results may have deviated based on this. That being said, our final results only perform slightly worse than the previous literature.

Tasin's final best-performing model when using SMOTE synthetic oversampling used bagging. When using Adasyn for synthetic oversampling, XGBoost was also found by Tasin to be the best. Our work showed similar results, and the superiority of XGBoost. We can confidently say that our methodology was sound and reinforced previous studies.

3. Dataset and Evaluation

For this project, we propose to use the CDC's Diabetes Health Indicators Dataset. This dataset was collected by the CDC "to better understand the relationship between lifestyle and diabetes in the US" (Centers for Disease Control and Prevention, 2023a). The dataset is comprised of 253,000 patients with features in both health metrics, such as alcohol consumption, cholesterol, BMI, etc., and social/economic metrics, such as income, education level, or other wealth-based questions.

To get our train/test split, we can take this dataset and split it 80%/20% while still ensuring that a representative distribution of the non-diabetic and diabetic/pre-diabetic classes is present in each grouping. This will allow us to verify that all classes are accurately represented in each set, and the training set or testing set is not exceptionally biased towards one class. This is especially important for this dataset, as the diabetes/pre-diabetes classification only makes up about 14% of the dataset.

Our implementation of the split results in train/test having 202,944 and 50,736 examples, respectively, an 80%/20% split. Within each subset, we also have a class balance similar to the entire dataset. Train has 28,311 positive classes and Test has 7,035 positive classes. In total for the dataset, there are 218,344 negative cases and 35,346 positive cases.

Because we are dealing with potentially life-threatening health issues, the best metric we can use is recall. Optimizing recall means minimizing false negatives. False negatives should be weighted much higher than false positives in our case, as a false negative could mean that this disease could go undiagnosed until it is too late for a patient. Risking this is unnecessary, and training our model to be conservative with its predictions could save lives. Accuracy would not be a suitable metric due to the large class imbalance in our dataset.

Hyperparameter	Values
Penalty	None, L1, L2, ElasticNet
Solver	LBGFS, Liblinear, Newton-cg, Newton-Cholesky, Sag, Saga
Max Iterations	50, 100, 150

Table 1. Logistic regression possible hyperparameters

Hyperparameter	Values
Criterion	Gini, Entropy
Max Depth	4,5,6,7,8,9,10,11,12,15,20,30,40,50,70,90,120

Table 2. Decision Tree possible hyperparameters

4. Methods

Because we are outputting a binary decision, a simple baseline method to implement would be logistic regression. Using this statistical approach, we can determine how a non-learning classifier would make very quick predictions on our data. Additionally, we determined how well the linearity assumption for our model holds, i.e., whether a linear combination of our input features can calculate our output. Poor results from logistic regression signified that there are non-linearities in the data.

As many of these features are binary features, we decided to implement a decision tree as a good basic ML model. This would allow us to capture some non-linearities in the data, as well as provide us with an interpretable model. With our high-dimensional data, the model handles splitting only where most important, reducing overfitting.

For each of our methods, we began by separating the features from their targets, and splitting this into an 80%/20% train/test split as described in 3. We decided to include all 21 given features, but rely on the models to regularize for features that are more or less important. For each respective model, we then determined hyperparameters by optimizing the recall on the development sets after fitting the models to training data. Our predictions were then compared to the true classifications of the data, and we then generated classification reports and interpreted them.

4.1. Logistic Regression

For Logistic Regression, we chose the hyperparameters based on recall performance in the dev set. We tested all available solver algorithms, as well as their regularization methods (we will call "penalty"). The resulting best solver is the LBFGS solver paired with no penalty. The LBFGS solver method is a method that updates the model parameters based on the average of the gradients for each data point. Another hyperparameter we used was a maximum number of iterations of 200, a value found by trying various values to attempt to maximize recall.

Hyperparameter	Values
1st Hidden layer size	25,50,100
2nd Hidden layer size	25,50,100
3rd Hidden layer size	None, 25,50,100

Table 3. Neural network possible hyperparameters

Hyperparameter	Values
Minimum child weight	0.2, 1
Gamma	0.1, 0.25, 0.5, 1
Subsample	0.3, 0.6, 0.8
Colsample by tree	0.4, 0.7
Max depth	3, 5, 7

Table 4. XGBoost possible hyperparameters

4.2. Decision Tree

For our Decision Tree, we chose the hyperparameters based on recall performance from 5-fold cross-validation. Through various testing, we found hyperparameters for impurity decrease, criterion, max tree depth, and example weights. To start, we used scikit-learn's GridSearchCV to test various max tree depths from 4-150 and different criterion methods, 'gini' and 'entropy'. This method tested every combination, and only returned the best hyperparameters based on the stated goal of recall. Because of our class imbalance, we also increased the weight of the positive classes, in order to have more importance on the decisions of the model. To do this, we tested various class weights, starting with just a basic inverse relationship to the prevalence of the class in the dataset. We iterated through values surrounding the simple inverse relationship, and found that weighting the positive classes more than just their inverse would lead to better recall for the model.

4.3. Neural Network

For our Neural Network, we used a Multi-layered perceptron with two to three hidden layers. We again chose the hyperparameters based on recall performance from 5-fold cross-validation with grid search on all possible combinations of hyperparameters. For the neural network, our hyperparameters were the size of the three hidden layers. We also included class weights to counteract the class imbalance, equalizing their effects on the weights.

4.4. XGBoost Tree

For our XGBoost Tree, we searched over many possible hyperparameters, including max depth, min child weight, subsample rate, and others. Just as before with decision tree, we were able to weight the positive class higher than its imbalanced level. For our final model, we chose values that resulted in parity in the final weighting between both classes. An interesting effect of the weighting choice was the relationship between accuracy and recall. With the XGBoosted tree, we could keep increasing the weighting of the positive class, constantly improving the recall, but at the expense of our f1-score and accuracy. While recall was our main metric, classifying all points as positive is a trivial way to get a good recall. For this reason, we opted for a value that would greatly increase our recall at only slight detriment to the accuracy and f1-score. We found that value to be parity.

All discovered hyperparameters will be discussed in the following section, 5.

5. Experiments

5.1. Logistic Regression

We first ran our logistic regression model to get a sense of our baseline performance, using the LBFGS solver and no regularization. As discussed previously, our preferred evaluation metric is recall, as we aim to minimize false negatives. This turned out to be an important distinction to make. To choose hyperparameters, we ran `sklearn`'s `GridSearchCV`, searching through parameters including which penalty and solver to use, as well as the maximum iterations. Our final results were a max iteration of 200, no penalty, and `lbfgs` as our solver.

Both the test and development sets had 86% accuracy but a recall of only 15%, a stark contrast. To visualize our data, we plotted our test set results into a confusion matrix.

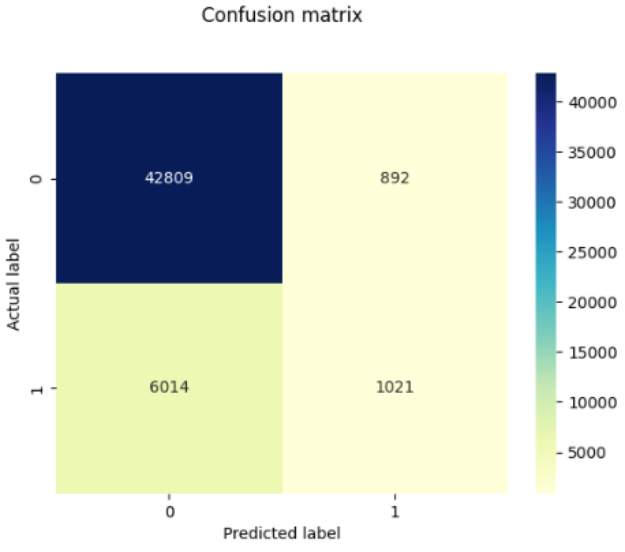

Figure 1. Confusion matrix representing the logistic regression predictions made from our test set. The actual label (0 representing no diabetes and 1 representing prediabetes/diabetes) is on the y-axis and the model output label is on the x-axis

From Figure 1, we can see that the model performs very

poorly on patients with a diabetes diagnosis, with 6014 of those with diabetes classified without it, while only 1021 patients with diabetes correctly classified. We can see that there is a large scope for improvement.

Another insight we can obtain from our logistic regression model is a glimpse at feature importance gleaned from the coefficients of each of our features.

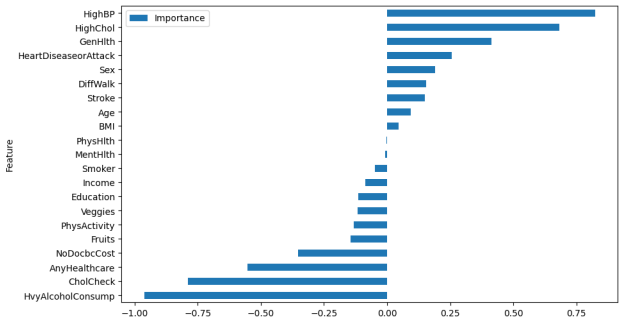

Figure 2. Feature importance based on the coefficients of each feature in the linear regression model. A more positive coefficient indicates a stronger correlation, while a more negative coefficient indicates a stronger anticorrelation.

From Figure 2, high blood pressure and high cholesterol strongly correlate to diabetes, which generally aligns with the medical consensus on the risk factors for diabetes. Interestingly, heavy alcohol consumption has a very negative coefficient, meaning that it is anticorrelated with diabetes, which is odd considering heavy alcohol consumption is a health risk.

Another way to look at the feature importance is through permutation importance. In this method, each feature is shuffled, and the increase in loss is determined. The features that cause the highest loss when randomly shuffled are the ones that are ”important” to the model’s predictor.

In this case, BMI is deemed to have the largest importance by far, which is a different result than the coefficients. This may be because BMI ranges from 15-30, while the many binary features are 0 or 1, meaning that the coefficient for BMI is smaller for the same importance. In this way, evaluating different metrics is useful to find insights about the data. As we delve into more computationally-heavy models, we can use these results for feature selection to reduce complexity.

5.2. Decision Tree

The next model we explored was the decision tree. To determine hyperparameters, we used grid search again with 5-fold cross-validation, choosing between Gini or Entropy criteria and a 4-150 max depth. We kept the minimum

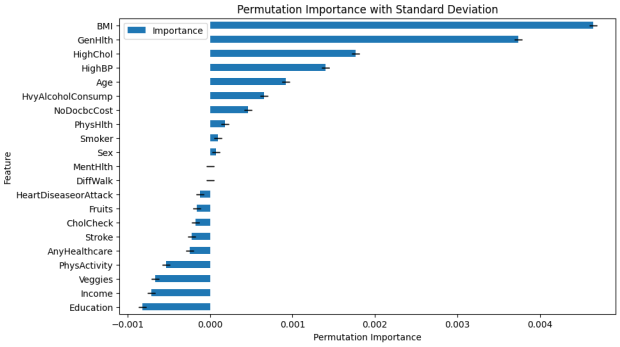

Figure 3. Permutation importance of each feature in the logistic regression model. The average of 10 repeats was plotted along with bars representing the standard deviation.

impurity decrease at 0 to avoid bias towards a negative classification, as we are aiming to avoid false negatives. We chose Gini and a max depth of 50 due to the highest cross-validation recall, as seen in Figure 4.

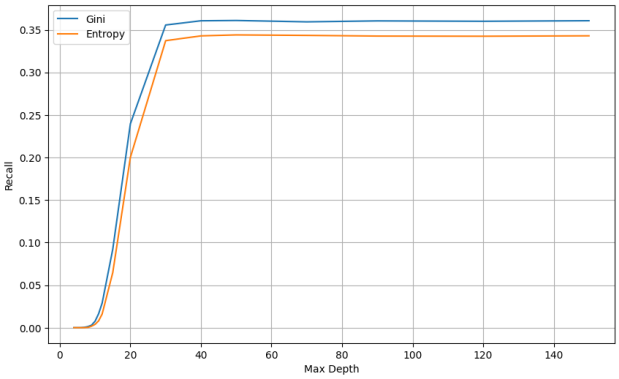

Figure 4. Cross validation recall of the decision tree when varying the max depth and using Gini and entropy as the criteria. Max depth of 50 and Gini had the highest recall and were thus chosen as the parameters for the model that we ran on the test data

While we got a test accuracy of 79%, which is lower than logistic regression, our test set recall was higher, at 34.9%. The increase is promising, as it shows that tree-based models may be a path for future exploration, for example, by using random forests. The many binary features may make tree-based models, such as the decision tree, more suitable for this problem.

5.3. Neural Network

We wanted to create deep learning methods to approach this problem, so we started with neural networks. We again implemented grid search, searching both for 2 and 3 layer neu-

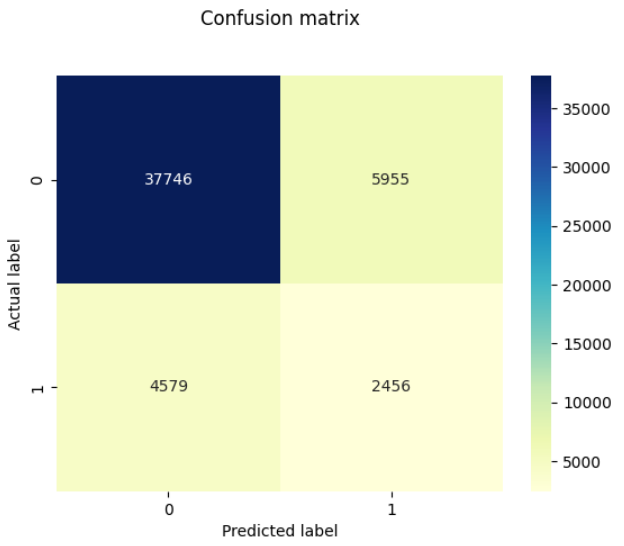

Figure 5. Confusion matrix representing the decision tree predictions made from our test set. The actual label (0 representing no diabetes and 1 representing prediabetes/diabetes) is on the y-axis and the model output label is on the x-axis.

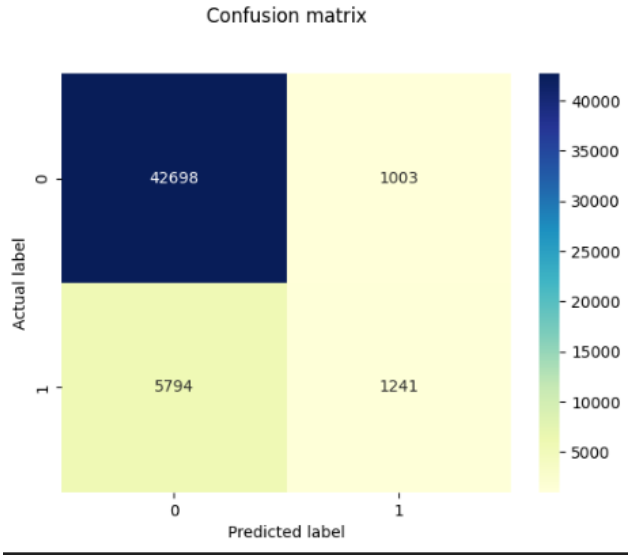

Figure 6. Confusion matrix representing the neural network predictions made from our test set. The actual label (0 representing no diabetes and 1 representing prediabetes/diabetes) is on the y-axis and the model output label is on the x-axis.

ral networks, each with each permutation of any value from `[25, 50, 100]`. We then changed the class weights to equalize the impact of the positive and negative classes on the model. Max iterations had to be cut to 30, as higher values took significant times to finish. The best hyperparameters we found were a 3 layer model with hidden layer sizes of `[25, 100, 25]`. This resulted in an accuracy of 87%, better than any other model, but a recall of only 18%. This signaled to us that this method most likely does not need further exploration, as we have already seen a decision tree classifier perform better than this method.

As we can see in the following confusion matrix, the model is significantly biased towards classifying datapoints as negative. This is extremely problematic for a health scenario, and demonstrates that this model should not be fit for real-world use.

5.4. XGBoost

The last method we chose stemmed from the previous promise that was shown through our decision tree classifier. XGBoost is a method to generate an ensemble of decision trees. It takes samples of the dataset and builds shallow trees for each one, which are used for an ensemble decision. We again used grid search to find our ideal parameters, deciding between parameters like max tree depth, gamma loss reduction values, minimum weight for a new child, subsampling ratio for each column in each tree, and more. The final hyperparameters were as follows:

	Logistic Regression	Decision Tree	Neural Network	XGBoost
Accuracy	86%	79%	87%	74%
Recall	14%	36%	18%	75%

Table 5. Test set accuracy and recall from best-performing model of each category

```
{'colsample_bytree': 0.4, 'eta':
0.001, 'gamma': 0.1, 'lambda': 2,
'max_depth': 3, 'min_child_weight':
0.2, 'subsample': 0.3}
```

Our final results supported our hypothesis that this would be a good method to explore. We obtained a greater recall than any other method, though at a slight cost to the accuracy. Our final recall was 0.75 and accuracy 0.74. The following confusion matrix shows how the predictions were distributed. As the chart shows, we have an extremely low number of false negatives, our goal, but our false positives are quite high. If this method returned that a patient was positive, there would be a twice as likely change that they were actually negative rather than positive. While this is much better than a false negative, we must admit this is a limitation of the model.

6. Discussion

While the XGBoost performed better than the LR and Decision tree, it is still not high enough to be used in a task

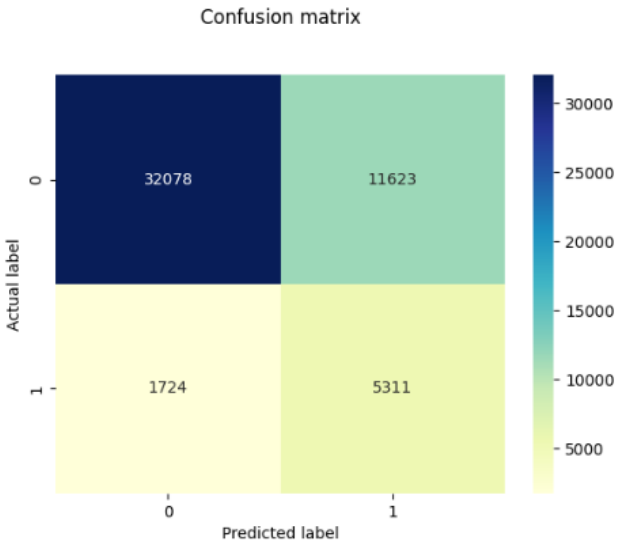

Figure 7. Confusion matrix representing the XGBoosted tree predictions made from our test set. The actual label (0 representing no diabetes and 1 representing prediabetes/diabetes) is on the y-axis and the model output label is on the x-axis.

High BP	High Cholesterol	Cholesterol Check	BMI
1	1	1	23
Smoker	Stroke	Heart Dis. or Attack	Phys. Activity
0	0	0	1
Fruits	Veggies	Heavy Alc. Consumption	Healthcare
1	1	0	1
No Doctor bc. of Cost	Gen. Health	Mental Health	Physical Health
0	3	0	0
Difficulty Walking	Sex	Age	Education
0	0	8	5
Income			
5			

Table 6. True Negative from Decision Tree Classifier

such as diabetes detection where false negatives should be avoided. Because of this, our current models do not provide a robust enough method to safely diagnose diabetes or pre-diabetes in patients. One reason for this may be complicated relationships between features that our current models cannot detect well. For example, here are cases of a True Negative, False Negative, True Positive, and False Positive. These datapoints were chosen from model predictions on our final Decision Tree Classifier. Each of the following values from each datapoints in the tables corresponds to these features: ['High BP', 'High Cholesterol', 'Cholesterol Check', 'BMI', 'Smoker', 'Stroke', 'Heart Disease or Attack', 'Physical Activity', 'Fruits', 'Veggies', 'Heavy Alcohol Consumption', 'Any Healthcare', 'No Doctor because of Cost', 'General Health', 'Mental Health', 'Physical Health', 'Difficulty Walking', 'Sex', 'Age', 'Education', 'Income'].

True Negative: Table 6

High BP	High Cholesterol	Cholesterol Check	BMI
1	1	1	20
Smoker	Stroke	Heart Dis. or Attack	Phys. Activity
1	1	0	1
Fruits	Veggies	Heavy Alc. Consumption	Healthcare
1	1	0	1
No Doctor bc. of Cost	Gen. Health	Mental Health	Physical Health
0	5	30	30
Difficulty Walking	Sex	Age	Education
1	0	11	6
Income			
3			

Table 7. False Negative from Decision Tree Classifier

High BP	High Cholesterol	Cholesterol Check	BMI
0	0	1	26
Smoker	Stroke	Heart Dis. or Attack	Phys. Activity
0	0	1	1
Fruits	Veggies	Heavy Alc. Consumption	Healthcare
0	0	0	1
No Doctor bc. of Cost	Gen. Health	Mental Health	Physical Health
0	3	0	5
Difficulty Walking	Sex	Age	Education
1	0	13	6
Income			
6			

Table 8. True Positive from Decision Tree Classifier

In this example, we can see the patient has a low BMI, checks their cholesterol levels, and has a good overall physical health, leading the model to believe that the patient is, correctly, non-diabetic.

False Negative: Table 7

In this example, the patient has a low BMI, but has high blood pressure, cholesterol, difficulty walking, and a poor general health. Through model text visualization, it is apparent that BMI is taken as important to the model early, so it has a great effect on the prediction. This is the most dangerous diagnosis, because a False Negative could undiagnose a patient that might have a potentially life-threatening disease. The model needs to be able to consider all of these points together to generate a safe diagnosis.

True Positive: Table 8

This patient shows difficulty walking, heart disease, and does not eat any fruits or veggies, leading the model to correctly classify them as diabetic or pre-diabetic.

False Positive: Table 9

This patient has many health problems, high blood pressure,

High BP	High Cholesterol	Cholesterol Check	BMI
1	1	1	27
Smoker	Stroke	Heart Dis. or Attack	Phys. Activity
0	0	0	1
Fruits	Veggies	Heavy Alc. Consumption	Healthcare
0	1	0	1
No Doctor bc. of Cost	Gen. Health	Mental Health	Physical Health
0	2	1	0
Difficulty Walking	Sex	Age	Education
0	0	8	6
Income			
8			

Table 9. False Positive from Decision Tree Classifier

cholesterol, BMI, but they do not have any heart issues, they stay active and eat healthy. The first part of this patient description leads the model to classify them as positive though they are not. For a safe model, this is ok on occasion, as it makes conservative estimates for the health of the patients.

We can possibly improve this in the future through feature scaling in order to improve training for models such as our Logistic Regression. With some features like BMI and General Health being scaled to a value significantly higher than binary features, they might affect the training of these models. For a visualization, see 8. This is also important to be able to put many features on a level playing field, as some binary features might be more important than some larger scale features. For example, High Blood Pressure might be more important than Income, but Income is scaled much larger.

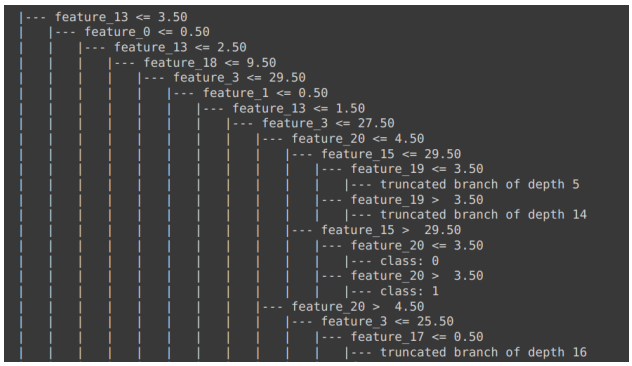

Figure 8. Text visualization of early branching of decision tree model. Important features, such as BMI (feature_3) and General Health (feature_13) show up many times, showing importance of these highly-scaled features.

7. Conclusion

Over the past semester, we have made good progress on tackling our problem of diabetes detection. We made a logistic regression model to establish a baseline and gather insights on feature importance. We were also able to implement a baseline ML model, the decision tree, which gave us better performance in our chosen metric, recall. Additionally, since the mid-semester report, we successfully implemented two more complex models, a neural network and XGBoost. We also determined the optimal hyperparameters for each model to improve performance through methods such as grid search.

Since the midterm report, our recall has significantly grown from a maximum at 36% using the decision tree. Moving from our decision tree to XGBoosted Tree method doubled our recall to 75%, further supporting our hypothesis that tree-based methods were promising. Future research could explore other architectures or ensembles between different types of models.

Through this project, we learned that different machine learning models, can perform at massively different levels. Most surprising was that our deep-learning method proved significantly worse than even a simple decision tree in the metric of recall, and was only slightly better in accuracy.

References

Centers for Disease Control and Prevention. CDC Diabetes Health Indicators. In *UCI Machine Learning Repository*. 2023a. URL https://archive.ics.uci.edu/dataset/891/cdc+diabetes+health+indicators.

Centers for Disease Control and Prevention. What is Diabetes? 2023b. URL https://www.cdc.gov/diabetes/basics/diabetes.html.

I. Tasin, T. U. Nabil, S. I. R. K. Diabetes prediction using machine learning and explainable ai techniques. *Healthc Technol Lett.*, 10((1-2)):1–10, 2022.

M. K. Hasan, M. A. Alam, D. D. E. H. M. H. Diabetes prediction using ensembling of different machine learning classifiers. In *IEEE Access*, volume 8, pp. 76516–76531, 2020.

A. Appendix

All code can be found at the following GitHub repository: https://github.com/Will-Dolan/CS467-finalproject

The dataset can be accessed at https://archive.ics.uci.edu/dataset/891/cdc+diabetes+health+indicators