Diabetes Predictor Report

★ Introduction

Data set: https://www.kaggle.com/datasets/tanshihjen/early-stage-diabetes-risk-prediction

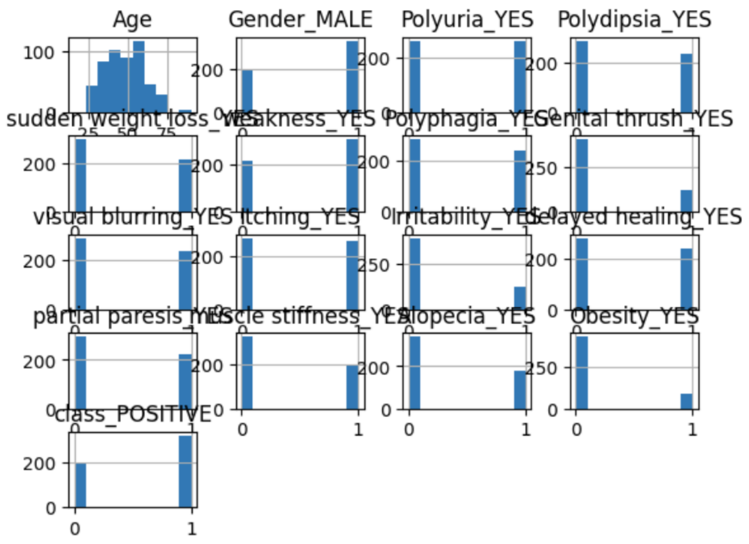
The dataset I chose shows different predictors/symptoms of diabetes, then says whether the person actually had diabetes or not. It has 17 columns which includes predictors like age, gender, sudden weight loss, and obesity. Of course the label, called “class” in this dataset, is one of the columns as well. The dataset has 520 rows/cases. Most of the feature columns are booleans, with either a yes or no if the person had that symptom or not. My goal is to use these predictors to predict whether or not someone of that description has diabetes or not. Given that KNN and neural networks performed well in the previous test, I’ll try using them again. And this time, instead of using decision trees, I’ll try using random forests.

★ Data Cleaning

The data was already pretty clean, so there wasn’t much to do. I started off by guessing the types of each column. To do this, I used the same function I used in HO4 and HO5. This step is important because the columns needed to be labeled before I ran functions like scale_data(), which only works on numerical columns. A lot of the cleaning process involved running my data through the classifier, noticing the accuracy was low, theorizing about why the accuracy was low, taking more steps to clean the data, and then repeating. The main changes I made were converting all the boolean columns to integer columns with 1 representing yes, and 0 representing no. The reason why I did this is because I realized that the classifier isn’t very effective on boolean columns, and is instead much more accurate with integer values. Another step of the cleaning process was scaling the age column. Since every other column was an integer column representing 1/0 for yes/no, the values in the age column were relatively very large, making them seem like a very significant feature. I noticed that when I scaled the age column to be between 0 and 1 (the same range as the other columns) the accuracy of the classifier went up by a lot.

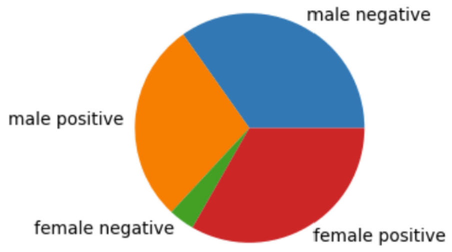
★ Data Visualization

Again, the first visualization I made was a histogram of the overall (cleaned) data. Although *I* don’t need to understand the meaning of the data in order to make predictions, it is still interesting to look at and learn from.



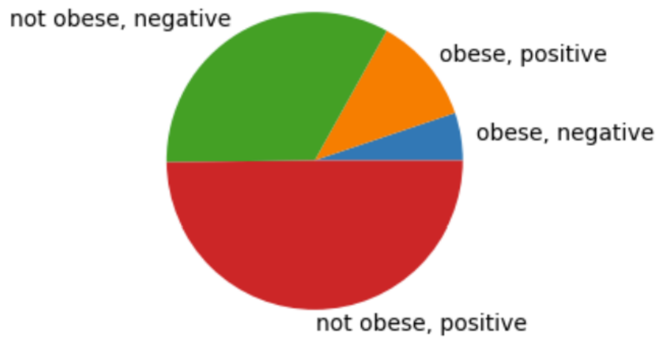
Based on these histograms, we can see that most of the participants in this dataset are ages 40-60, and also that most of the participants were diagnosed with diabetes. Since I know that this doesn't reflect real world data (in the real world it is only a small population of the total that have diabetes) I can tell that this data is not a random set, and thus might not account for the amount of people that have these symptoms but don't have diabetes.

The next thing I was curious about was whether there was a correlation between sex and diabetes. In order to make this visualization, I found the number of rows that returned a 1 for the sex_MALE column and also a 1 for class_POSITIVE (this correlates to the male positive section in the pie chart), and repeated that process for male negative, female negative, and female positive. Then I turned this data into a series so that pandas could create the chart.



We can clearly see that there were more male participants in the dataset than female, and also that of the people recorded, males were almost half and half with positive and negative, while most females in the dataset were positive. It is *possible* that this means that females are more likely to have diabetes; but I can't be 100% because we're not sure how the dataset was chosen.

The next feature I wanted to look at was obesity. Based on my previous knowledge of diabetes, I thought that there would be a strong correlation between obesity and diabetes, but the data seemed to prove this assumption wrong. And again, in order to create this visualization I found the number of rows that returned 1 for Obesity_POSITIVE and 1 for class_POSITIVE to find the number of (obese, positive) values, and repeated the process to get the other three values.



Although we're not sure about how the dataset was chosen, this visualization makes it quite clear that there are a lot of people who have diabetes that aren't obese, while the number of people that fit in the (obese, positive) category is relatively small.

★ Modeling

The three classifiers I chose included KNN, random forest, and neural networks. I played around with the parameters used for all the classifiers, and landed on settings that seemed to optimize the classifier. For KNN, I found that k = 3 neighbors, max features of 1, and n estimators of 10 maximized the average accuracy of the folds. For the random forest, a max depth of 20 seemed to work the best. And for neural networks, and alpha of 1, and a max iteration of 1,000 produced the best results. For both the random forest and the neural network I used a random state of 42, to get random, but controlled results.

In terms of accuracy the random forest performed the best by far, reaching a mean accuracy of 98.1%, and it was also the most precise/consistent model, with a standard deviation of accuracy of 0.00860. The KNN classifier was the runner up, with a mean accuracy of 95.3%, and surprisingly the neural network was the least accurate, with a mean accuracy of 91.7%. However overall, the differences in accuracy were not statistically significant, so all the models still performed well.

Model	Mean Accuracy	Standard Deviation of Accuracy
KNN	0.953	.0176
Random Forest	0.981	.00860

Neural Network	0.917	.0188

★ Analysis

I was surprised by these results; based on how the previous test went, I expected a higher accuracy from KNN and especially neural networks. I did hear that random forests are generally more accurate and consistent than decision trees so I wasn't too surprised that it had a better accuracy than my previous decision tree classifier. I wonder if the drop in accuracy was because the data in this set were mostly 0s and 1s, instead of having a wide range of numbers like my previous dataset? And I wonder if the reason why the random forest outperformed the KNN and neural networks so much was because random forests are better when there are binary values like 0 and 1, instead of continuous values like the values in the previous dataset. I'm not sure what I could do to improve the accuracy of the models, besides experimenting more with the parameters. For the KNN classifier, I wasn't quite sure what the max features and n estimators did, so I didn't tweak those values as much. So if I were looking to improve the accuracy of my model, that would probably be the first thing I would look into.