Comparing Classification Models

Ahmad Sai

Overview

For this project, I decided to explore how different classification models compare and perform when applied to medical scenarios, such as stroke prediction, heart attack prediction, or diagnosing a patient with diabetes, which was what I decided to focus my models on in this project. In the real world, diagnosing a patient with any kind of medical condition, such as diabetes, requires extensive lab and blood work as well as numerous other tests that are performed by medical professionals, but I decided to see how a mathematical model can perform in terms of accurately diagnosing patients.

A large part of my projection included having me explore a multitude of different classifications models that produce an output of, 0 or 1, true or false, yes or no, diabetic or not diabetic, etc. The most prominent and well-known models that I came across while researching this topic were; Logistic Regression models, Support Vector Machine (SVM) models, Decision tree models, and Extreme Gradient boosting (Xgboost) models. All of these models are used for classification problems, which was what my project, and many other medical applications, truly are in their most basic form.

Now that I knew what models I will be using in my project it was time to find a dataset on which I can implement these models and compare their accuracy. The dataset that I used is found on Kaggle and it contains data about people for which I can use to predict if a person is diabetic or not.

Introduction

According to the CDC approximately 37.3 million Americans have diabetes, which equates to 1 in every 10 people having diabetes. What is more interesting is that 1 in 5 people with diabetes don't know they have it, that is truly scary! Can a model be built where body features, e.g. BMI, and lab/blood work be input and an accurate diagnosis for diabetes be outputted? We'll find out soon enough! Classification models are used to solve decision problems that normally have binary output, such as diabetic or not diabetic. To see how accurate the most industry-used classification models are when their uses are applied to medical applications, I decided to build 5 different classification models and compare their accuracy relative to the data.

Data Cleaning

Using the dataset I found on Kaggle, I decided to import that data into Python and look for any anomalies that may stand out. At first glance, everything seems to check out, the data set has 2000 rows and 9 columns.

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	2	138	62	35	0	33.6	0.127	47	1
1	0	84	82	31	125	38.2	0.233	23	0
2	0	145	0	0	0	44.2	0.630	31	1
3	0	135	68	42	250	42.3	0.365	24	1
4	1	139	62	41	480	40.7	0.536	21	0

After further investigation, I realized some columns had values that don't make sense such as "0" in the "SkinThickness". Having these data points in my data frame will throw off the models I create and therefore decrease their accuracy. Thus, I decided to clean the dataset and remove these junk rows. After cleaning the data I only had 1035 rows remaining, which was just about enough to build my models.

To see how far spread out the data really was, I used the .std() function in Pandas to view the standard deviations of the data. As we can clearly see on the left, the data is very spread out and this can cause abnormality when training the models, therefore I normalized the data with the largest value in each column and the result was a much better spread of the data, which we can clearly see on the right.

```
Pregnancies                 3.167531
Glucose                    30.653888
BloodPressure              12.336324
SkinThickness              10.558989
Insulin                   111.489069
BMI                         7.097899
DiabetesPedigreeFunction    0.332292
Age                        10.047212
Outcome                     0.468827
dtype: float64
```

```
Pregnancies                 0.186325
Glucose                     0.154818
BloodPressure               0.112148
SkinThickness               0.167603
Insulin                     0.149851
BMI                         0.105781
DiabetesPedigreeFunction    0.137311
Age                         0.124040
Outcome                     0.468827
dtype: float64
```

Figure 1: Standard deviations of the dataset. Before & After

Age Group Demographics

To visualize the demographic of people that were in the dataset, I decided to graph the number of people in each age group, the number of diabetics in the group, and the percentage of the group that was diabetic.

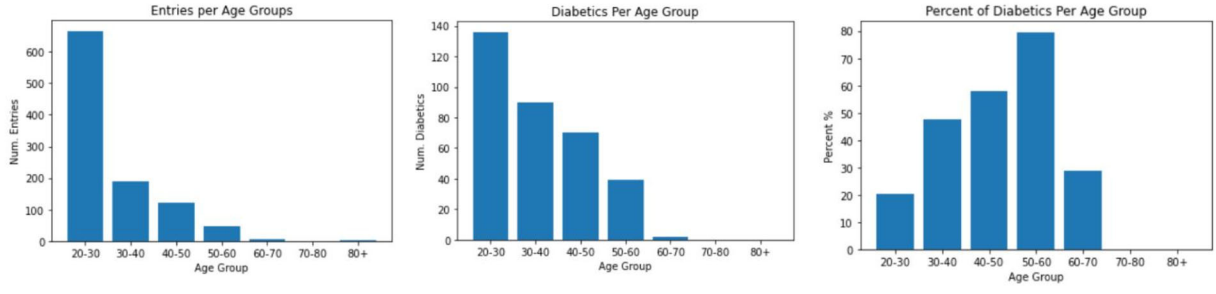

Figure 2: Visual representation of patient demographics

As we can clearly see from the first plot, there are a lot of people in this dataset that fall within the 20-30 age range, with fewer people in each group as the age increases. From the second plot we can interpret that

the age group with the largest number of diabetics is the 20-30 age range. This is to be expected as there are more entries for this age range than other age ranges. What is intriguing is the percentage of diabetics per age group, we can clearly see in the 3rd plot that the 20-30 age range has the lowest ratio of diabetics, and the 50-60 age group has the highest. This is interesting as its bar plot seems to follow a normal distribution.

Preparing the Data

In this step, I split the dataset into 2 pieces, 80% for training, and 20% for testing, but before doing that I shuffled the data 100 times to prevent an uneven split of diabetics and non diabetics in the training and testing sets.

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
442	0.235294	0.590909	0.581818	0.428571	0.161290	0.494784	0.095041	0.296296	0.0
544	0.058824	0.444444	0.709091	0.460317	0.102151	0.476900	0.150826	0.358025	0.0
1685	0.058824	0.404040	0.672727	0.174603	0.080645	0.447094	0.217769	0.271605	0.0
1189	0.117647	0.878788	0.800000	0.587302	0.161290	0.663189	0.266942	0.296296	1.0
445	0.000000	0.909091	0.709091	1.000000	0.018817	0.885246	1.000000	0.308642	1.0
...	...	...	...	...	...	...	...	...	...
777	0.000000	0.691919	0.618182	0.222222	0.198925	0.369598	0.059091	0.259259	0.0
127	0.058824	0.595960	0.527273	0.571429	0.126344	0.496274	0.107851	0.283951	0.0
1190	0.117647	0.535354	0.509091	0.428571	0.221774	0.432191	0.176033	0.271605	0.0
1120	0.529412	0.732323	0.727273	0.730159	0.174731	0.564829	0.263223	0.493827	1.0
125	0.058824	0.444444	0.272727	0.666667	0.133065	0.819672	0.204959	0.320988	1.0

828 rows × 9 columns

Figure 3: Training Set

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
1278	0.000000	0.818182	0.690909	0.888889	0.134409	0.792846	0.313636	0.308642	1.0
313	0.176471	0.570707	0.454545	0.158730	0.114247	0.439642	0.258678	0.308642	0.0
1582	0.000000	0.686869	0.672727	0.777778	0.295699	0.299553	0.338843	0.543210	1.0
1892	0.294118	0.702020	0.581818	0.555556	0.188172	0.426230	0.169835	0.320988	0.0
1549	0.117647	0.419192	0.590909	0.444444	0.088710	0.548435	0.259917	0.296296	0.0
...	...	...	...	...	...	...	...	...	...
1193	0.000000	0.636364	0.781818	0.428571	0.161290	0.408346	0.212810	0.259259	0.0
1733	0.117647	0.792929	0.672727	0.555556	0.591398	0.587183	0.055372	0.370370	0.0
421	0.117647	0.474747	0.618182	0.285714	0.102151	0.387481	0.231818	0.259259	0.0
956	0.058824	0.702020	0.563636	0.650794	0.645161	0.606557	0.221488	0.259259	0.0
309	0.117647	0.626263	0.618182	0.444444	0.275538	0.490313	0.361570	0.370370	1.0

207 rows × 9 columns

Figure 4: Testing Set

Building the models

Now that the data is clean, prepared, and ready to go, all I have to do is build the different classification models and train and test them on their respective data sets! The first model I built was a Logistic Regression model that used the columns: Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, and Age to predict the patient's diabetics diagnostics. I named this model the 'Default Logistic Regression' model because it used all the variables to predict the outcome. The 'Default Logistic Regression' model results in a prediction accuracy of 73.9%, which is pretty good, but not good enough for applications that could result in life or death.

The next model I built was also based on Logistic Regression, but this time I specifically picked the input variables based on some research I did to see what the leading causes of diabetes are. The columns I used for this 'Modified Logistic Regression' were: Glucose, BloodPressure, Insulin, BMI, and DiabetesPedigreeFunction. I had initially thought that because I specifically picked variables that tend to have more of an impact on whether a person is diabetic or not that the 'Modified Logistic Regression' would be more accurate, but unfortunately it wasn't. The 'Modified Logistic Regression' only achieved a prediction accuracy of 72.95%, which is less than the 'Default Logistic Regression' model.

Now it was time to use models that are industry standard for many applications, the first of which was the 'Support Vector Machine (SVM)' model. The SVM model achieved a prediction accuracy of 76.33%, this is slightly better than both of the previous Logistic Regression models, but it is still not good enough. The next model I built is based on Decision Tree algorithms. This model takes in data, performs a tree-based algorithm, and gives an output very quickly and efficiently. The Decision Tree model had an accuracy of 95.17% which is very good. The last model to test out was the Extreme Gradient boosting (Xgboost) model, and surprisingly enough this model performed better than the Decision Tree model, achieving total prediction accuracy of 95.65%.

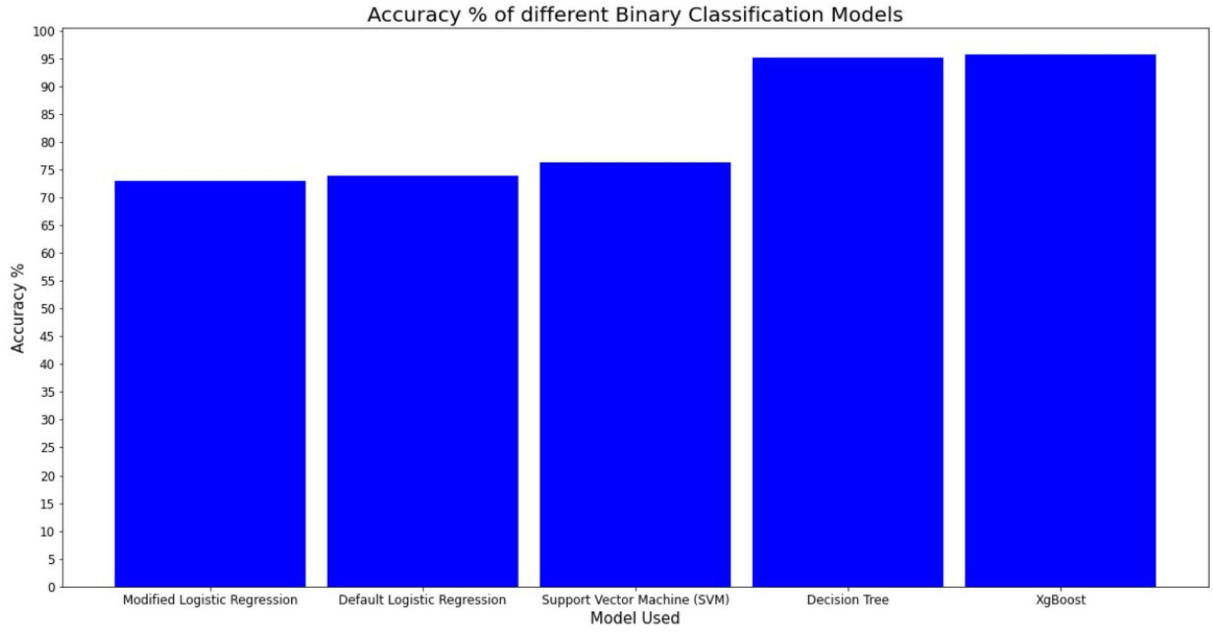

Figure 5: Visual Representation of the Accuracy of Each Model

Conclusion

In conclusion, it seems like each classification model has its drawbacks and limitiations negatives. For example, the logistic regression models were relatively accurate, but not accurate enough to be used in a medical setting. It seems pretty obvious that industry-standard classification models will perform many times better than a simple logistic regression model, and this was the case for Xgboost and the Decision Tree model. The Xgboost and Decision tree models performed outstandingly well, both achieving a prediction accuracy of over 95%. What stood out to me was the SVM model; I was anticipating a better prediction accuracy for this model because of its wide use in the Data Science industry, but unfortunately, that wasn't the case in this application.

What conclusions can we draw from this? Firstly, a model can have a high prediction accuracy in theory but this will not always be the case in real life. We can't gamble the lives of patients on computer models when there are trained and experienced medical professionals who do a much better job at diagnosis patients Secondly, this data set wasn't all that large, with 2000 rows before cleaning and 1035 rows after. Maybe if the dataset was much larger, then the accuracy of these models will decrease because the data will statistically be more diverse. Lastly, these models can be considered 'toy' models, as most of the time an incredible amount of statistical work goes into preparing the data before feeding it into the model, and often times these models are tweaked specifically for the application they are to be used in. The models used in this project are what we call 'vanilla', they are purely for testing and no meaningful conclusions can be drawn from them.

Reflection

Overall, this project was a ton of fun! I learned a lot about cleaning data and different classification models. I also gained a lot of valuable skills in Pandas and Sklearn that I will definitely put to use in future data science projects. One thing, that I could have done was build GUI, where the user can input values for the variables: Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, and Age, to see whether this hypothetical 'person' they inputted will be diagnosed with diabetes based on the classifcaiton model they choose. Another addition I would have done in this project if I had more time, was build these same models on a different dataset, e.g. Heart Attack Prediction dataset, and compare how they perform on the different medical conditions that the models are trying to predict. One thing that I had trouble with in this project was finding the right dataset. At first, I was planning on doing this project on a 'Stroke Prediction Dataset' but a large number of missing values in that dataset led me to spend countless hours on Kaggle looking for the right dataset to use in this project.