INTELLIGENT SYSTEMS FOR DIABETES DIAGNOSTICS

By: Alexis Castellanos

JUNE 21, 2021

CIS 479

Overview

This project aims to create a chatbot system based on a reengineering version of my rule-based system which handles diagnosis of diabetes in patients. The full chatbot diagnosis script will contain two main parts. The first portion will be the computational engine that is responsible for diagnosing a patient with diabetes. The second portion is an inference system for the user to input patient available data and receive a real time diagnosis. We will be using the dataset provided by National Institute of Diabetes and Digestive and Kidney Diseases named PIMA Indian Diabetes to train and test our model.

Inference Strategy Described:

The tools used for this project are python 3 under an Anaconda Jupyter Notebook environment. The pandas python module will be used to create all data frames used. The computational model will be built from tools found in sklearn's library as a tuned bagging decision tree. The chatbot system will be constructed from a deep neural network using the library TensorFlow. The pre-processing for the chatbot will be using NLTK's libraries. The chatbot system will use natural language processing to extract biometrics form the user. These biometrics (features) will then be translated to the computational model to predict a diabetes diagnosis, based on the provided features.

The biometrics the chatbot system will be asking for are the following:

Figure 1.0

Feature	Description	Unit of Measurement
Pregnancies	Number of times pregnant	Numeric
Glucose	Blood Sugar concentration over 2 hours in *Oral Glucose Tolerance Test*	mg/dL
Blood Pressure	Diastolic blood pressure test	mm Hg
Skin Thickness	Bodyfat skin fold test (triceps)	mm
Insulin	Serum insulin levels (2 hours)	mu U/ml
BMI	Body mass index Formula: (kg (weight)/ meters (height)2)	Kg/m(2)
Diabetes Pedigree Function	Likelihood if diabetes based on family history	Percent Probability
Age	In years	Numeric

Data and Pre-Processing

Computational Model:

The computational model will be trained on the PIMA Indian Diabetes Data set. This public data contains 768 total labeled examples, 500 diabetic and 268 non-diabetics. However, it is important to note some features in our data are recorded to be zero. Hence, patients may present up to 8 the features but not required. To overcome missing features in data points we may **impute** the missing values. Imputing is the process of inferring the missing data from the known portion of the data. Figure 2.0 displays the data set used for training and validating our computational engine.

Figure 2.0

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Diabetes
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

Chatbot Model:

The chatbot model will be trained on a json file created by me. This json files when paired with python contain a hash like properties and are very useful for storing and pulling data. The file contains eight tags that correspond to our established biomarkers seen in Figure 2.0. Inside each tag we can find corresponding examples of *language patterns* we expect a user to use when referring to biometric. This will be paired with a response the chatbot model will use when identifying a context. Responses are provided to the user as a reassurance of correct biomarker identification. On the development portion, this was used to see what tag needed more examples for training. Figure 2.1 below displays a screenshot of the json file used for training our chatbot model.

Figure 2.1

```
{"intents": [

        {"tag": "Pregnancies",
         "patterns": ["I have 2 kids", "I alot of children", "I have seven kids", "four children", "Pregnant five times","Never have been pregnante"],
         "responses": ["Great, thank you for informing me with your pregnancies"],
         "context_set": ""
        },
        {"tag": "Glucose",
         "patterns": ["Glucose Level is 100", "Glucose level is 200", "I dont know my glucose level","Sugar is 89","Blood sugar is 40","My blood sugar levels is 90"],
         "responses": ["Great, thank you for informing me with of your glucose levels"],
         "context_set": ""
        },
        {"tag": "BloodPressure",
         "patterns": ["Blood Pressure is 70", "Blood pressure level is 130", "I dont know my blood pressure"],
         "responses": ["Thank you for your blood pressure readings"],
         "context_set": ""
        },
        {"tag": "SkinThickness",
         "patterns": ["My skin thicness is 36mm", "I dont know my skinthicness", "skin thickness 23","Skin 90","Skin thick is 20mm","Skin is 30","Skin is 30mm","Skin is 80mm","The thicnkess of my skin is 40mm","I cant measure the thickness of my skin"],
         "responses": ["Thank you for your information on skin thickness "],
         "context_set": ""
        },
        {"tag": "Insulin",
         "patterns": ["My insulin is measured at 39", "Insulin level is unknown", "Insulin 35", "I dont know my insulin","My insulin levels are 50","Insulin 60","Insulin 40","50 is my insulin"],
         "responses": ["Thank you for infomring us with your insulin levels"],
         "context_set": ""
        },
        {"tag": "BMI",
         "patterns": ["My BMI is about 30", "I cannot find my BMI", "bmi is 23","According to the chart value my bmi is 30","My bmi is 18","BMI reading is 40","I am overweight at 33 BMI on scale"],
         "responses": ["Thank you for providing us with your BMI"],
         "context_set": ""
        },
        {"tag": "DiabetesPedigreeFunction",
         "patterns": ["I dont know my pedigree value", "my diabetes pedigree value is 0.6", "My pedigree value is 0.1", "i dont know my diabetic family history"],
         "responses": ["Thanks for providing us with your diabetes pedigree value"],
         "context_set": ""
        },
        {"tag": "Age",
         "patterns": ["I am 24 years old", "I am 30", "Age is 43","Age 70","Age is none","Age","AGE","Years of age","Years old"],
         "responses": ["Thank you for proving us with your age"],
         "context_set": ""
        }
    ]
}
```

The preprocessing steps for converting our natural language training data start off by tokenizing each example found in the patterns list. **Tokenization** is the process by which text is divided into sub portions (words) called tokens. These tokens are the building blocks of pattern recognition in natural language and considered the base step for our next process, stemming. Each word token may now be stemmed, **stemming** is the process of reducing a word down to their root form.

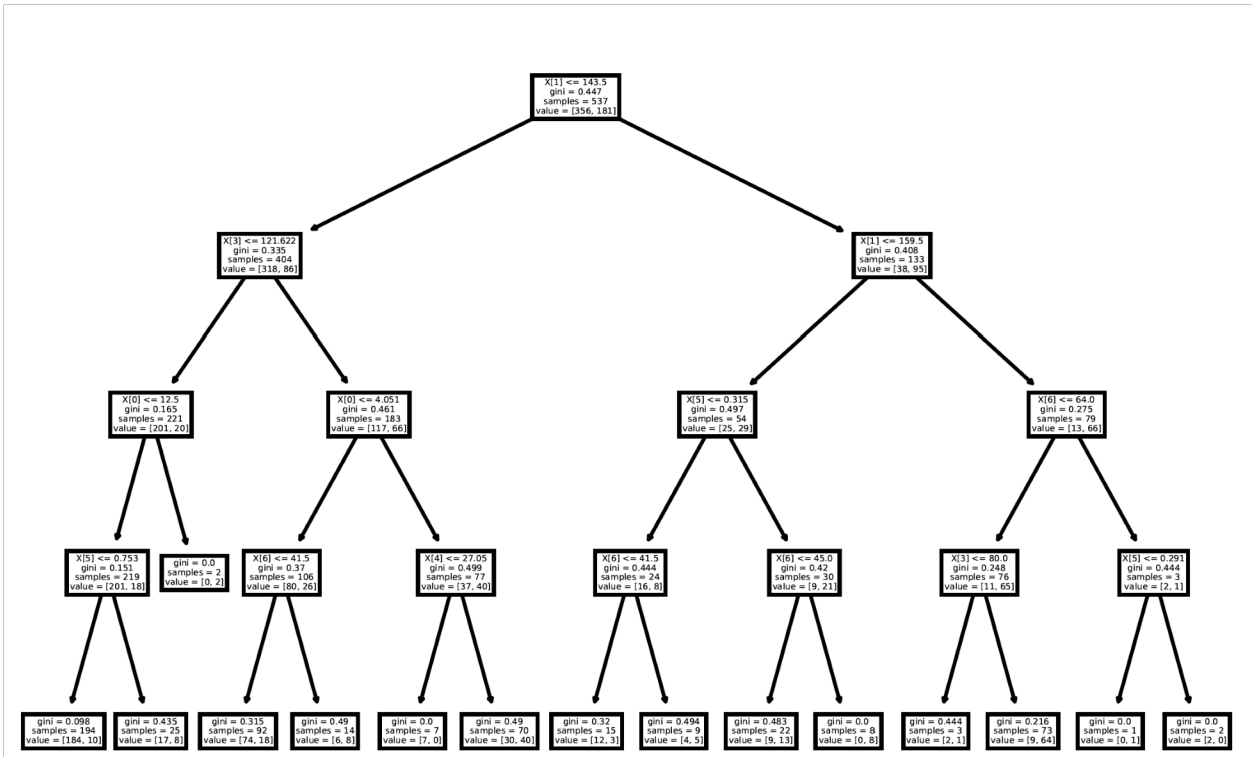
Now we may use a **bag of words** method to give numeric frequencies of tokens (stemmed) and create a corresponding array for each tag (biometric). This process will only apply to our training data and does not happen during the testing phase. Given that we have now created arrays to feed into our model, we can proceed to the architecture portion on this report.

Model Architectures and Performance

Computational Model:

As mentioned before, this project is based on a reengineering version of my rule-based system which handles diagnosis of diabetes in patients. The original version was a base decision tree with an accuracy of 70%. The structure of this base tree can be seen in figure 3.0, while the contraction and results can be seen on figure 3.1.

Figure 3.0



Decision Tree

Figure 3.1

To the right we can see the results of our base system. The decision tree created is a binary tree we used to create our rule-based system. In order to increase the accuracy score of my computational system, I decided to hyper tune the decision tree under a bagging technique.

```
from sklearn import tree
from sklearn.metrics import confusion_matrix
tree_model = tree.DecisionTreeClassifier()
tree_model.fit(X_train, y_train.ravel())

y_pred = tree_model.predict(X_test)
print("Accuracy: ",tree_model.score(X_test,y_test))
confusion_matrix(y_test,y_pred)
```

```
Accuracy:  0.70995670995671
```

The new and improved model used a bagging decision tree classifier, and its parameters (number of estimators max samples and random state) were decided upon testing trails. A Bagging classifier is a meta-estimator that fits base classifiers (decision tree) each on random subsets of the original dataset and then aggregate their individual predictions (either by voting or by averaging) to form a final prediction. The hyper parameter **oob_score** has been set to True because it may help estimate errors by generalizing given cases. Due to the size of this model, its structure cannot be printed however a generalized depiction of this process can be seen in figure 3.2. This model has performed to an accuracy score of 90% as seen figure 3.3 below.

Figure 3.2

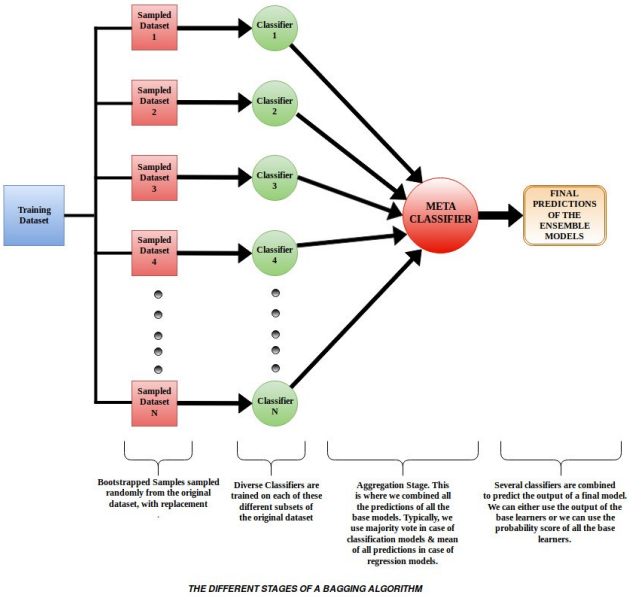


Figure 3.3

```
bagging = BaggingClassifier
(DecisionTreeClassifier(random_state=42),
 n_estimators=500,max_samples=100,
bootstrap=True,n_jobs =1,random_state=42,
oob_score= True)

bagging.fit(x_train,y_train)

print(classification_report(bagging.predict(x_validate),(y_validate))
```

```
              precision    recall  f1-score   support

           0       0.94      0.91      0.93       102
           1       0.84      0.88      0.86        52

    accuracy                           0.90       154
   macro avg       0.89      0.90      0.89       154
weighted avg       0.90      0.90      0.90       154
```

Chatbot Model:

The chatbot model will be constructed using TensorFlow and its supported module tflearn. This will be a deep learning model with 5 total layers. The first layer is the input layer, it is responsible for setting the shape [0,81] of our neural network. The second and third layers are hidden layers each containing 8 neurons to match the number of biomarkers we are using. The following layer is our activation layer using softmax activation to assign the probability of a biomarker (tag) given a user sentence. The last layer is our output layer utilizing regressing to provide a singular output with the highest context probability. We can see a diagram of this DNN (top portion is cut off) in the next page figure 3.4 and its corresponding code in figure 3.5.

Figure 3.4

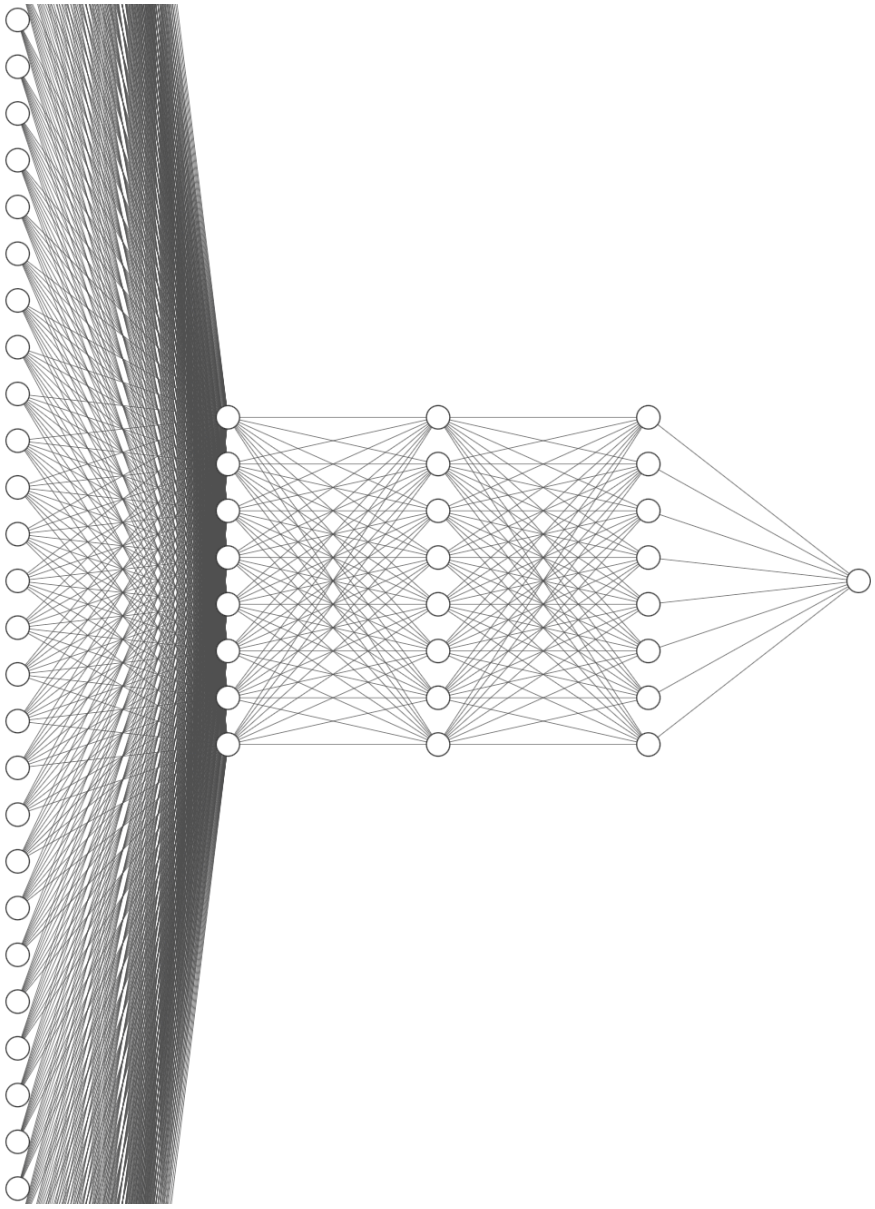


Figure 3.5

```
def build_nn():

    net = tflearn.input_data(shape=[None, len(training[0])])
    net = tflearn.fully_connected(net, 8)
    net = tflearn.fully_connected(net, 8)
    net = tflearn.fully_connected(net, len(output[0]), activation="softmax")
    net = tflearn.regression(net)
    model = tflearn.DNN(net)

    return model
```

```
def train_nn():
    model = build_nn()
    model.fit(training, output, n_epoch=1000, batch_size=8, show_metric=True)
    model.save("chatbot_diabetes2.tflearn")

    return model
```

Chatbot Conversation – Positive Case

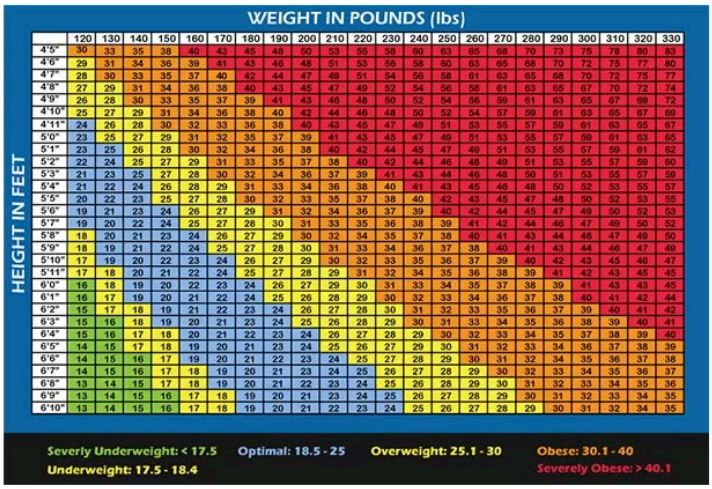
The screen recording below is of a conversation with the chatbot. The chatbot can correctly extract the biometric values to diagnose a user. The Chatbot is able to use its computational engine to accurately diagnose the diabetic patient and communicate it back to him/her.

****Right click and press play to view real time demo conversation. ****

```
In [*]: chat()
```

```
Hello! this is Sugerbot, I can people determain if they are in risk of diabetes.
All you have to do, is provide me with the following information and type done to for me to calculate:
```

	Bio Markers	How to measure
0	Pregnancies	Number of times pregnant
1	Glucose	Plasma glucose concentration over 2 hours in a...
2	BloodPressure	Diastolic blood pressure (mm Hg)
3	SkinThickness	Triceps skin fold thickness (mm)
4	Insulin	2-Hour serum insulin (mu U/ml)
5	BMI	Please view chart below
6	DiabetesPedigreeFunction	function which scores likelihood of diabetes b...
7	Age	Age (years)




```
You:
```

```
In [ ]:
```

Chatbot Conversation – Negative Case

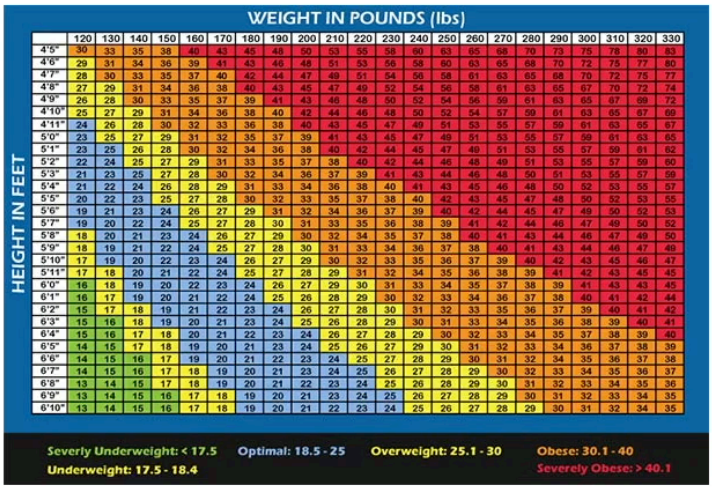
In this example we can see the chatbot does not need to be given the information by order, does not require all biometrics to be provided and is able to correctly diagnose the negative diabetes patient. However, as we know our computational model has scored at 90% accuracy, therefore our chatbot can only diagnose at best 90%. The chatbot will give a list of diabetic warning symptoms / conditions.

****Right click and press play to view real time demo conversation. ****

```
In [*]: chat()
```

```
Hello! this is Sugerbot, I can people determain if they are in risk of diabetes.
All you have to do, is provide me with the following information and type done to for me to calculate:
```

	Bio Markers	How to measure
0	Pregnancies	Number of times pregnant
1	Glucose	Plasma glucose concentration over 2 hours in a...
2	BloodPressure	Diastolic blood pressure (mm Hg)
3	SkinThickness	Triceps skin fold thickness (mm)
4	Insulin	2-Hour serum insulin (mu U/ml)
5	BMI	Please view chart below
6	DiabetesPedigreeFunction	function which scores likelihood of diabetes b...
7	Age	Age (years)




```
You:
```

```
In [ ]:
```

To conclude, the performance of the base model to the bagging model was an increase of 20% accuracy. While the chatbot system seems to grasp the correct context with its limited 44 lines of training data. This combination creates a well performing and cohesive system, however there is much more room for improvement. With the addition of more training data examples and development we could improve this case and even apply this same concept to diagnose other common diseases. Common heath oriented techwear, such as smart watches, could be used as inferences between users and diagnosis. The data collected from these devices can be very useful for training and diagnosing in the future, I hope to be a part of.