Early Stage Diabetes Risk Prediction using the Bernoulli Naïve Bayes Classifier

Supervised Classification algorithm to determine the diabetic risk of an individual as a binary output, given 16 independent flags.

University of Bath
Jayanth Vasanth Kumar
July-August 2021

Introduction

This report aims to document a supervised machine learning algorithm to analyse and train on the given data set on diabetic characteristics in order to classify new data points, given a set of limited parameters or flags, to effectively identify if the individual is of high or low risk of diabetes/diabetic disorders. **Disclaimer**: The developed algorithm, and subsequently this report, is by no means approved, endorsed or tested accurate by any official medical governing body or organisation, and merely aims to showcase machine learning models tested on real world data to those who are interested, and/or for educational purposes only.

Information on Diabetes

Diabetes Mellitus, or more commonly referred to as simply 'Diabetes', is a form of long-term (usually lifetime) metabolic disorder which often results from consuming large amounts of sugar-rich foods over a long period of time; in other words, categorised by historically high blood sugar levels in the tested individual. Three main types of diabetes are known to exist: *Type 1*, *Type 2* and *Gestational diabetes,* each with varying levels of commonality in the human population. The types are described below:

Type 1 (T1D): Previously known as 'juvenile diabetes[1]', is a type where the pancreas inherently fails to produce a sufficient amount of insulin due to the loss of the insulin synthesis cells - 'beta cells'. Studies suggests that T1D is predominantly caused genetically[2] (with some evidence that lifestyle also plays a minor role). T1D diabetic people are usually required to inject themselves manually with insulin on a regular daily basis.

Type 2 (T2D): This is by far the most common type of diabetes with up to 95% of the diabetic population in the US having type 2 (according to the US National Diabetes Statistics Report released in 2020, 10.5% of the entire population are recorded to have diabetes[3]). T2D is primarily caused by high-sugar lifestyles and is when the individual's cells fail to respond naturally to insulin from a gained insulin resistance in the cells. Major factors which lead to diagnosis include insufficient exercise and excessive body weight.

Gestational Diabetes: The third and final type is relatively uncommon and is when a high blood sugar level is recorded during/as a result of pregnancy from either

[1] https://en.wikipedia.org/wiki/Diabetes

[2] https://www.healthline.com/health/diabetes/is-type-1-diabetes-genetic

[3] https://www.cdc.gov/diabetes/pdfs/data/statistics/national-diabetes-statistics-report.pdf

insulin resistance or lack of insulin production. This type affects around 3-9%[4] of pregnancies.

While some symptoms may be unique to a certain type of diabetes, the majority of symptoms which are valid across all types are among the following:

- Regular thirstiness
- Frequent urination (especially during the night)
- Tiredness, Weight and muscle-mass loss
- Genital itchiness, bouts of thrush
- Changes in vision, blurriness

About the Dataset and Problem

The data collected to form this dataset is a joint effort from four members in the UK and Bangladesh; researchers from Queen Mary University London and Metropolitan University Sylhet respectively (the dataset source webpage and full references with contact information of the researchers is located in the '*References*' section of the report - page ___). According to the webpage, the entirety of the data is accumulated through direct questionnaires handed to real patients registered at Sylhet Diabetes Hospital in Sylhet, and is approved by a doctor (although a formal proof of approval has not been published as of the date of writing of this report).

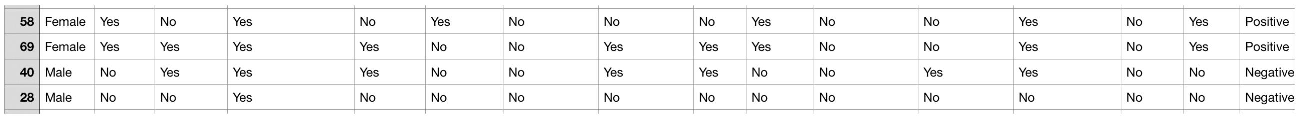
Now, taking a look at the dataset, there are precisely 520 entries of data, each with 17 attributes, resulting in a product of 8,840 data points and making this a *multivariate* dataset. Another point worth mentioning is the fact that this dataset contains empty or nullified values - possibly from the patient missing out an entry point in the questionnaire, or from another reason leading to an empty result. Here are the different attributes listed below, with their data type class:

Attribute name	**Data type**	**Description**
Age	INTEGER (Positive values, i.e. 43)	Age of patient in years
Sex	STRING (Binary classification, i.e. Male or Female)	Gender identification of patient
Polyuria	STRING (Binary classification, i.e. Yes or No)	Abnormally large volume of dilute urine production
Polydipsia	STRING (Binary classification, i.e. Yes or No)	Abnormal thirstiness for no apparent reason; as a symptom for disease

[4] https://en.wikipedia.org/wiki/Gestational_diabetes

Attribute name	**Data type**	**Description**
Sudden weight loss	STRING (Binary classification, i.e. Yes or No)	Loss of weight in the patient within a relatively short period of time
Weakness	STRING (Binary classification, i.e. Yes or No)	General weakness in the patient
Polyphagia	STRING (Binary classification, i.e. Yes or No)	Abnormally large appetite; as a symptom for disease
Genital thrush	STRING (Binary classification, i.e. Yes or No)	Thrush in the patient's genital area
Visual blurring	STRING (Binary classification, i.e. Yes or No)	Abnormal blurring in the patient's eyesight
Itching	STRING (Binary classification, i.e. Yes or No)	Excessive local or global itchiness in the patient
Irritability	STRING (Binary classification, i.e. Yes or No)	General Irritability in patient
Delayed healing	STRING (Binary classification, i.e. Yes or No)	Abnormally slow time for patient to heal wounds
Partial paresis	STRING (Binary classification, i.e. Yes or No)	Condition of muscular weakness in the patient as a result of nervous damage
Muscle stiffness	STRING (Binary classification, i.e. Yes or No)	Abnormal stiffness in the muscles
Alopecia	STRING (Binary classification, i.e. Yes or No)	Complete or partial loss of hair, baldness
Obesity	STRING (Binary classification, i.e. Yes or No)	Patient BMI calculated as over 30
Class (Outcome)	STRING (Binary classification, i.e. Positive or Negative)	Final outcome; positive or negative case of Diabetes

Here is an extract taken directly from the dataset:

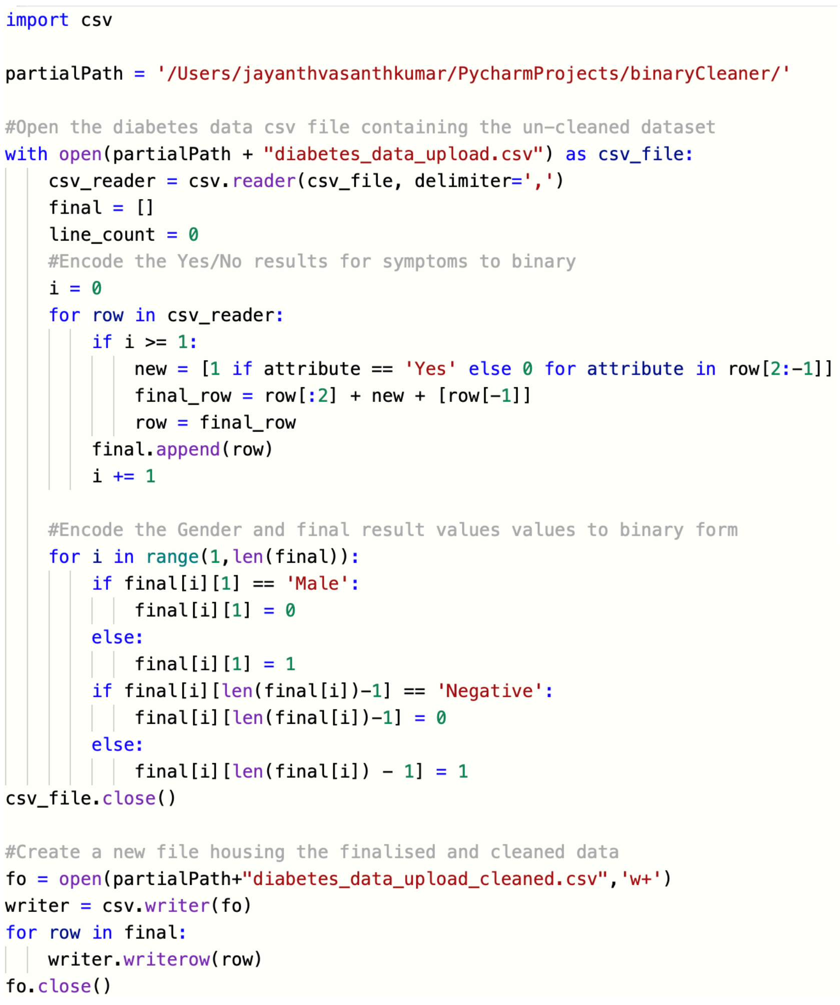


58	Female	Yes	No	Yes	No	Yes	No	No	No	Yes	No	No	Yes	No	Yes	Positive
69	Female	Yes	Yes	Yes	Yes	No	No	Yes	Yes	Yes	No	No	Yes	No	Yes	Positive
40	Male	No	Yes	Yes	Yes	No	No	Yes	Yes	No	No	Yes	Yes	No	No	Negative
28	Male	No	No	Yes	No	No	No	No	No	No	No	No	No	No	No	Negative

A problem we have here is that much of the dataset consists of binary categorical string values such as 'Yes', 'Male' or 'Positive'; feeding this directly into the training process of the Naïve Bayes will likely lead to errors and learning incorrect patterns. To solve this, a process called *categorical encoding* is employed to translate the strings into numerical data, which is in turn easier for the model to process (new columns are synthesised, each categorical attribute having $\lceil \log_2(n) \rceil$ columns, where n is the total number of different categories). In most cases, categorical data consists of more than

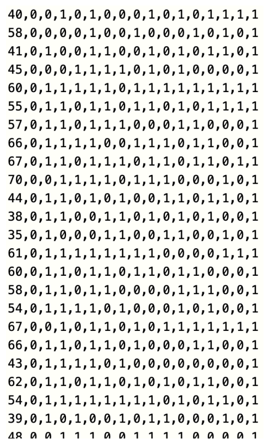
2 categories, however in our case we only have a maximum of 2 (binary) categories; no extra columns required since $\lceil \log_2(2) \rceil = 1$. This is solved by simply encoding each binary class with either 1 or 0 respectively: Yes=1; No=0 or Male=0; Female=1 etc.

A short python script can be developed to enforce categorical encoding; the script is shown below along with the input and output data.



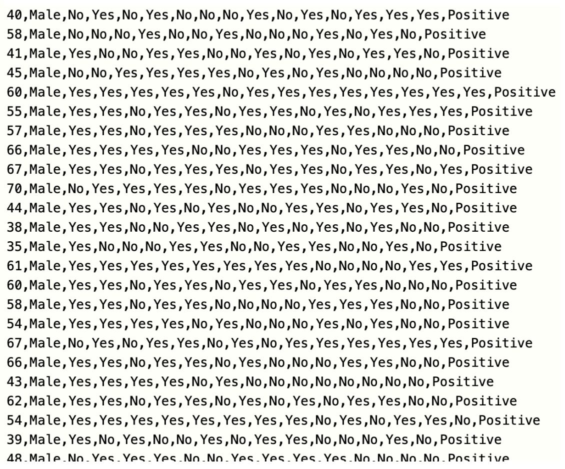
```python
import csv

partialPath = '/Users/jayanthvasanthkumar/PycharmProjects/binaryCleaner/'

#Open the diabetes data csv file containing the un-cleaned dataset
with open(partialPath + "diabetes_data_upload.csv") as csv_file:
    csv_reader = csv.reader(csv_file, delimiter=',')
    final = []
    line_count = 0
    #Encode the Yes/No results for symptoms to binary
    i = 0
    for row in csv_reader:
        if i >= 1:
            new = [1 if attribute == 'Yes' else 0 for attribute in row[2:-1]]
            final_row = row[:2] + new + [row[-1]]
            row = final_row
        final.append(row)
        i += 1

    #Encode the Gender and final result values values to binary form
    for i in range(1,len(final)):
        if final[i][1] == 'Male':
            final[i][1] = 0
        else:
            final[i][1] = 1
        if final[i][len(final[i])-1] == 'Negative':
            final[i][len(final[i])-1] = 0
        else:
            final[i][len(final[i]) - 1] = 1
csv_file.close()

#Create a new file housing the finalised and cleaned data
fo = open(partialPath+"diabetes_data_upload_cleaned.csv",'w+')
writer = csv.writer(fo)
for row in final:
    writer.writerow(row)
fo.close()
```

**Extract of output*



```
40,0,0,1,0,1,0,0,0,1,0,1,0,1,1,1,1
58,0,0,0,0,1,0,0,1,0,0,0,1,0,1,0,1
41,0,1,0,0,1,1,0,0,1,0,1,0,1,1,0,1
45,0,0,0,1,1,1,1,0,1,0,1,0,0,0,0,1
60,0,1,1,1,1,1,0,1,1,1,1,1,1,1,1,1
55,0,1,1,0,1,1,0,1,1,0,1,0,1,1,1,1
57,0,1,1,0,1,1,1,0,0,0,1,1,0,0,0,1
66,0,1,1,1,1,0,0,1,1,1,0,1,1,0,0,1
67,0,1,1,0,1,1,1,0,1,1,0,1,1,0,1,1
70,0,0,1,1,1,1,0,1,1,1,0,0,0,1,0,1
44,0,1,1,0,1,0,1,0,0,1,1,0,1,1,0,1
38,0,1,1,0,0,1,1,0,1,0,1,0,1,0,0,1
35,0,1,0,0,0,1,1,0,0,1,1,0,0,1,0,1
61,0,1,1,1,1,1,1,1,1,0,0,0,0,1,1,1
60,0,1,1,0,1,1,0,1,1,0,1,1,0,0,0,1
58,0,1,1,0,1,1,0,0,0,0,1,1,1,0,0,1
54,0,1,1,1,1,0,1,0,0,0,1,0,1,0,0,1
67,0,0,1,0,1,1,0,1,0,1,1,1,1,1,1,1
66,0,1,1,0,1,1,0,1,0,0,0,1,1,0,0,1
43,0,1,1,1,1,0,1,0,0,0,0,0,0,0,0,1
62,0,1,1,0,1,1,0,1,0,1,0,1,1,0,0,1
54,0,1,1,1,1,1,1,1,1,0,1,0,1,1,0,1
39,0,1,0,1,0,0,1,0,1,1,0,0,0,1,0,1
48,0,0,1,1,1,0,0,1,1,1,1,0,0,0,0,1
```

**Extract of input*



```
40,Male,No,Yes,No,Yes,No,No,No,Yes,No,Yes,No,Yes,Yes,Yes,Positive
58,Male,No,No,No,Yes,No,No,Yes,No,No,No,Yes,No,Yes,No,Positive
41,Male,Yes,No,No,Yes,Yes,No,No,Yes,No,Yes,No,Yes,Yes,No,Positive
45,Male,No,No,Yes,Yes,Yes,Yes,No,Yes,No,Yes,No,No,No,No,Positive
60,Male,Yes,Yes,Yes,Yes,Yes,No,Yes,Yes,Yes,Yes,Yes,Yes,Yes,Yes,Positive
55,Male,Yes,Yes,No,Yes,Yes,No,Yes,Yes,No,Yes,No,Yes,Yes,Yes,Positive
57,Male,Yes,Yes,No,Yes,Yes,Yes,No,No,No,Yes,Yes,No,No,No,Positive
66,Male,Yes,Yes,Yes,Yes,No,No,Yes,Yes,Yes,No,Yes,Yes,No,No,Positive
67,Male,Yes,Yes,No,Yes,Yes,Yes,No,Yes,Yes,No,Yes,Yes,No,Yes,Positive
70,Male,No,Yes,Yes,Yes,Yes,No,Yes,Yes,Yes,No,No,No,Yes,No,Positive
44,Male,Yes,Yes,No,Yes,No,Yes,No,No,Yes,Yes,No,Yes,Yes,No,Positive
38,Male,Yes,Yes,No,No,Yes,Yes,No,Yes,No,Yes,No,Yes,No,No,Positive
35,Male,Yes,No,No,No,Yes,Yes,No,No,Yes,Yes,No,No,Yes,No,Positive
61,Male,Yes,Yes,Yes,Yes,Yes,Yes,Yes,Yes,No,No,No,No,Yes,Yes,Positive
60,Male,Yes,Yes,No,Yes,Yes,No,Yes,Yes,No,Yes,Yes,No,No,No,Positive
58,Male,Yes,Yes,No,Yes,Yes,No,No,No,No,Yes,Yes,Yes,No,No,Positive
54,Male,Yes,Yes,Yes,Yes,No,Yes,No,No,No,Yes,No,Yes,No,No,Positive
67,Male,No,Yes,No,Yes,Yes,No,Yes,No,Yes,Yes,Yes,Yes,Yes,Yes,Positive
66,Male,Yes,Yes,No,Yes,Yes,No,Yes,No,No,No,Yes,Yes,No,No,Positive
43,Male,Yes,Yes,Yes,Yes,No,Yes,No,No,No,No,No,No,No,No,Positive
62,Male,Yes,Yes,No,Yes,Yes,No,Yes,No,Yes,No,Yes,Yes,No,No,Positive
54,Male,Yes,Yes,Yes,Yes,Yes,Yes,Yes,Yes,No,Yes,No,Yes,Yes,No,Positive
39,Male,Yes,No,Yes,No,No,Yes,No,Yes,Yes,No,No,No,Yes,No,Positive
48,Male,No,Yes,Yes,Yes,No,No,Yes,Yes,Yes,Yes,No,No,No,No,Positive
```

Following the categorical encoding, the dataset is now placed into an abstracted form, ready to be fed into a model to be used as training.

The Naïve Bayes classifier

The Naïve Bayes model is a very simple, yet effective classification algorithm commonly used to classify a new entry given a set of features into a certain class (known as the '*maximum a posteriori*' estimate - this is expanded on later). In our case that would work out to classify a patient, given a set of 17 symptoms (if they are positive for each of them), as either positive (high risk) or negative (low/negligible risk) for diabetes. While there are several event models that could be used to model the distribution of the features, the *Bernoulli* Naïve Bayes model seems most sensible in our scenario given the nature of features being independent binary values. But first, we can state the essential Naïve Bayes equation - the identity from which all of the below propositions are developed from:

$$p(A\,|\,B) = \frac{p(B\,|\,A)p(A)}{p(B)}$$

Now, the determinant output class can be found by the Bayesian *Maximum A Posteriori* estimate:

$$\hat{c} = \underset{c\in\{0,1\}}{\arg\max}\ p(C = c\,|\,s)$$

Where c is the class and s represents the binary feature vector (symptoms) of the patient. Using Bayes' Theorem, the conditional probability can then be written as:

$$p(C = c\,|\,s) = \frac{p(s\,|\,C = c)p(C = c)}{p(s\,|\,C = 1)p(C = 1) + p(s\,|\,C = 0)p(C = 0)}$$

Since for both instances of C = 1 (Positive) and C = 0 (Negative) the denominator is common, we can state the following proposition:

$$p(C = c\,|\,s) \propto p(s\,|\,C = c)p(C = c)$$

$$p(C = c\,|\,s) = Kp(s\,|\,C = c)p(C = c),\ \forall K \in \mathbb{R}$$

We now need to identify the likelihood $p(s\,|\,C = c)$, relative to our model choice.

$$p(s\,|\,C = c) = \prod_{i=1}^{n} p_{ki}^{x_i}(1 - p_{ki})^{(1-x_i)}$$

Where p_{ki} is the deterministic probability of class c being a value $x_i \in \{0,1\}$. Substitution into the proportionality will give us:

$$p(C = c\,|\,s) = Kp(C = c)\prod_{i=1}^{n} p_{ki}^{x_i}(1 - p_{ki})^{(1-x_i)}$$

$$\therefore\ \hat{c} = \underset{c\in\{0,1\}}{\arg\max}\ [Kp(C = c)\prod_{i=1}^{n} p_{ki}^{x_i}(1 - p_{ki})^{(1-x_i)}]$$

Algorithm Decomposition and Analysis

```python
import numpy as np

from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import BernoulliNB
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.preprocessing import StandardScaler

import pandas as pd
from pandas.plotting import parallel_coordinates

import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap

#Reading the data from the csv, loading into a DataFrame object
dataset = pd.read_csv('diabetes_symptoms_dataset.csv')

#Segmenting the dataset into the patient features (Y) matrix and result (y) vector
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values

#Training to testing split set to 75:25 ratio
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)

sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

#Instantiate classifier object as a Bernoulli Naive Bayes model
classifier = BernoulliNB()

#===== FITTING PROCESS =====
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)

#TEST PATIENT FEATURE VECTOR; can be replaced for manual/custom input by admin
test_patient_symptoms = [99,1,0,0,1,1,0,0,0,0,0,0,0,0,0,0]
print("Preduction result: ",classifier.predict([test_patient_symptoms]))

#Instantiate confusion matrix of the trained model
matrix = confusion_matrix(y_test, y_pred)
print("Confusion matrix:\n",matrix)
print("\nAccuracy score: ",round(accuracy_score(y_test, y_pred),3)*100,"%")

#====== VISUALISAING =======
import seaborn as sns

#Utilising the lmplot method from the seaborn class to pass in the chosen parameters
#to generate the appropriate graph for binary features
#y-jitter of 0.11 chosen to spread out data points in the vertical axis such to improve
#the clarity of each of the class' population and density
sns.lmplot('Age', 'Polyuria', dataset, hue='class', fit_reg=False, y_jitter=(.11))
fig = plt.gcf()
fig.set_size_inches(10, 5)
plt.show()
```