

# Diabetes Prediction Based on Livelihood

Jacob Jojy  
Master of Computer Applications  
Amal Jyothi College of Engineering  
Kottayam, Kerala, India  
jacobjojy@mca.ajce.in

Binumon Joseph  
Master of computer Application  
Amal Jyothi College of Engineering  
Kottayam, Kerala, India  
binumonjosephk@amaljyothi.ac.in

**Abstract--** The purpose of this take a look at is to decide someone's diabetes risk relying on their lifestyle. there has been lots of development in the field of clinical prognosis using various gadget mastering algorithms. The peoples living in urban and Rural place have special existence. unique system gaining knowledge of algorithms had been used to forecast the risk of diabetes, and the outcomes were quite accurate, that's vital within the medical area. For databases, the performance of KN- Neighbours Classifier and Random wooded area Classifiers was decided to be the maximum accurate.

**Keywords--** Diabetes, Lifestyle, Urban and Rural areas, KN- Neighbours Classifier, Random Forest Classifiers

## I. INTRODUCTION

Diabetes influences extra than 30 million human beings international, with many extra at danger. To prevent diabetes and related health troubles, early prognosis and treatment are required. The purpose of this take a look at became to decide if someone is at hazard for diabetes primarily based on their way of life. Diabetes impacts extra than 30 million human beings global, with many more at danger.

Diabetes mellitus, normally called mellitus (DM), is a set of metabolic disorders characterised through continual high blood sugar ranges. excessive urination, continual thirst, and multiplied appetite are all symptoms of expanded glucose uptake. Diabetes, if no longer treated without delay, can lead to serious health issues, inclusive of diabetes ketoacidosis, hyperosmolar conditions, or maybe demise. this will cause lengthy-term side results which includes cardiovascular sickness, stroke, kidney failure, foot ulcers, and eye strain, among others.

previous research has checked out the effect of urbanization at the increase in diabetes on the regional / country wide stage. The reason of this study changed into to observe the hyperlink among urbanization and diabetes at the global level, as well as the function of intermediate variables (physical inactiveness, sugar use and weight problems).

The urban and rural way of life isn't like their manner of life. In rural areas they have got extra physical activity, less intake of junk meals, lower levels of strain in comparison to urban residing. although diabetes is likewise found in rural regions, it is much less common in people with diabetes than in people who live an city way of life. A observe carried out at the 2019 weight loss program of

people with Diabetes shows some indications of the connection between way of life and hazard for diabetes. sure, there are different causes of diabetes except life-style, looking for different causes beyond that may be prevented.

## II. RELATED JOB

Diabetes is a not unusual problem in India, with over 70% of adults suffering from this circumstance. distinctive students have used numerous techniques along with system studying and data mining to assume the signs of diabetes. just a few of them have used neural networks and genetic algorithms as nicely. due to the fact diabetes is a precautionary measure, many people have resorted to managed strategies that include device studying, statistics mining, and sensory networks.

The Pima Indians Diabetes Dataset (PIDD) has been used in many studies research to expect diabetes using mechanical gaining knowledge of strategies and the device device. gadget studying methods, statistics mining techniques, hybrid techniques, and neural network or genetic algorithms are a number of the various methods utilized by researchers.

## II. MATERIALS AND METHODS

- **Materials**  
The data collected for the analysis is collected from 2019 about the random peoples with different life styles.  
A descriptive study of urban and rural residential regions was given. The Klomargov-Smithera test was used to ensure that the variables were normal.

The following details are collected from the peoples:

- Age
  - Gender
  - Family Diabetes
  - Smoking and Alcohol habits
  - Sleep data
  - Junk food habit
  - Stress level
- K Nearest- Neighbor Classifier Algorithm
    - How the okay-Nearest Neighbor, Jr. (KNN) may be used to deal with the problems of drawing and separation yet used to address the demanding situations for the business divisions. Its predominant benefits are the ease with which it may be interpreted and the constrained time it takes to calculate. To calculate the distances among present statistics factors and each new data point, KNN makes use of the Euclidean distance feature. The k-Nearest Neighbor technique is based totally on a supervised gaining knowledge of approach and is one of the primary machine mastering algorithms. The okay-NN technique assumes that the brand new case / records and old cases are compared and places the new case in a category that is very much like the existing categories. The k-NN approach shops all available facts and classifies new statistics factors based on their similarity to previous records. which means that using the okay-NN approach, new information can be speedy taken care of right into a properly-defined class. despite the fact that the ok-NN approach may be used for each retransmission and modifying, it's miles most commonly used for isolation. The ok-NN algorithm is a non-parameter algorithm, which means that it does not make any assumptions approximately the records. it is also called a lazy pupil set of rules as it does now not study from the schooling without delay set; instead, it shops the database and uses it to cut up it over the years. Random woodland category algorithm.
    - The Random forest class forms a number of decision trees from the randomly decided on schooling database category. Votes from a few selection-making timber have been put together to set up a very last elegance of testing materials.
- "Random woodland is a software that mixes the range of decision trees in different sub-datasets and scales them to growth the expected accuracy of the database." in place of counting

on a single decision tree, the random forest collects predictions from each tree and predicts the final outcome primarily based on a couple of predictable votes. It takes much less time to teach compared to different algorithms. It as it should be predicts output and works rapid, even for huge datasets. although a large amount of records is lacking, it is able to nonetheless be true.

Step 1: select okay facts factors from the education set everywhere.

Step 2: For the statistics factors you have got selected, create subsets. Step three: pick out N for the wide variety of decision bushes you want to make.

Step four: undergo steps 1 and a couple of of again.

Step 5: Get predictions from every choice tree for brand spanking new data factors, and assign new records points to the maximum famous phase.

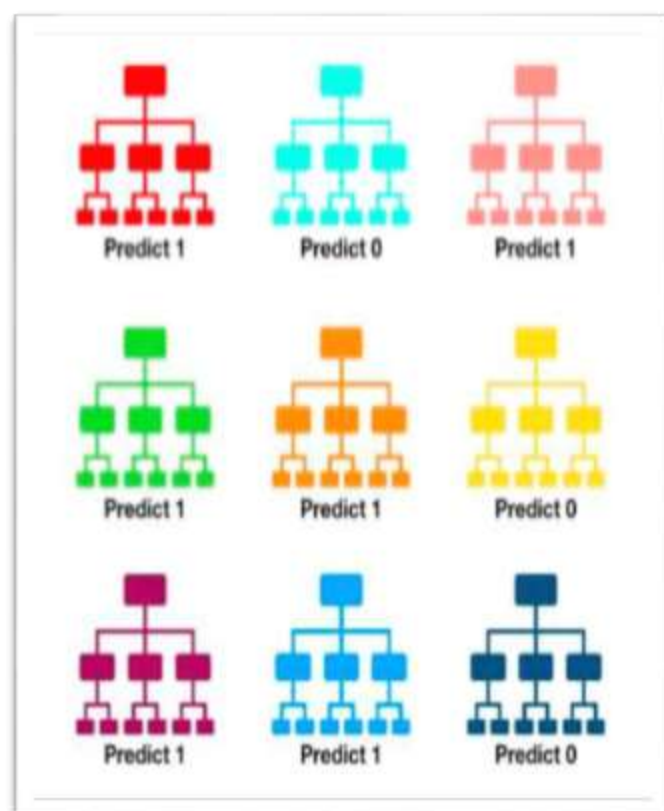


Figure-1 Random Forest Classification

The given algorithms are performed to make the data analysis using Jupyter.

### III. RESULTS

The algorithm performed on the given dataset given results.

Figure-2 Sample Dataset

```
In [3]: import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
import seaborn as sns
from sklearn.metrics import confusion_matrix, classification_report, f1_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import cross_val_predict, GridSearchCV
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
# Input data files are available in the read-only "../inputs/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the
import os
for dirname, _, filenames in os.walk('../Project/Amal/sample data'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

Figure-3 Importing Modules

```
In [4]: df = pd.read_csv(os.path.join(dirname, filename))
df.columns

Out[4]: Index: ['Age', 'Gender', 'Family_Diabetes', 'highBP', 'PhysicallyActive', 'Smoking', 'Alcohol', 'Sleep', 'InsulinUse', 'RegularMedicine', 'JunkFood', 'Stress', 'BPLevel', 'Pdiabetes', 'UrinationFreq', 'Diabetic'],
dtype: object

In [5]: df['BPLevel'].value_counts()

Out[5]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [6]: df['BPLevel'].value_counts()

Out[6]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [7]: df['BPLevel'].value_counts()

Out[7]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [8]: df['BPLevel'].value_counts()

Out[8]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [9]: df['BPLevel'].value_counts()

Out[9]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [10]: df['BPLevel'].value_counts()

Out[10]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [11]: df['BPLevel'].value_counts()

Out[11]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [12]: df['BPLevel'].value_counts()

Out[12]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [13]: df['BPLevel'].value_counts()

Out[13]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [14]: df['BPLevel'].value_counts()

Out[14]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [15]: df['BPLevel'].value_counts()

Out[15]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [16]: df['BPLevel'].value_counts()

Out[16]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [17]: df['BPLevel'].value_counts()

Out[17]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [18]: df['BPLevel'].value_counts()

Out[18]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [19]: df['BPLevel'].value_counts()

Out[19]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [20]: df['BPLevel'].value_counts()

Out[20]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [21]: df['BPLevel'].value_counts()

Out[21]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [22]: df['BPLevel'].value_counts()

Out[22]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [23]: df['BPLevel'].value_counts()

Out[23]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [24]: df['BPLevel'].value_counts()

Out[24]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [25]: df['BPLevel'].value_counts()

Out[25]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [26]: df['BPLevel'].value_counts()

Out[26]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [27]: df['BPLevel'].value_counts()

Out[27]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [28]: df['BPLevel'].value_counts()

Out[28]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [29]: df['BPLevel'].value_counts()

Out[29]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [30]: df['BPLevel'].value_counts()

Out[30]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [31]: df['BPLevel'].value_counts()

Out[31]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [32]: df['BPLevel'].value_counts()

Out[32]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [33]: df['BPLevel'].value_counts()

Out[33]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [34]: df['BPLevel'].value_counts()

Out[34]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [35]: df['BPLevel'].value_counts()

Out[35]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [36]: df['BPLevel'].value_counts()

Out[36]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [37]: df['BPLevel'].value_counts()

Out[37]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [38]: df['BPLevel'].value_counts()

Out[38]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [39]: df['BPLevel'].value_counts()

Out[39]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [40]: df['BPLevel'].value_counts()

Out[40]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [41]: df['BPLevel'].value_counts()

Out[41]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [42]: df['BPLevel'].value_counts()

Out[42]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [43]: df['BPLevel'].value_counts()

Out[43]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [44]: df['BPLevel'].value_counts()

Out[44]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [45]: df['BPLevel'].value_counts()

Out[45]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [46]: df['BPLevel'].value_counts()

Out[46]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [47]: df['BPLevel'].value_counts()

Out[47]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [48]: df['BPLevel'].value_counts()

Out[48]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [49]: df['BPLevel'].value_counts()

Out[49]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [50]: df['BPLevel'].value_counts()

Out[50]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [51]: df['BPLevel'].value_counts()

Out[51]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [52]: df['BPLevel'].value_counts()

Out[52]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [53]: df['BPLevel'].value_counts()

Out[53]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [54]: df['BPLevel'].value_counts()

Out[54]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [55]: df['BPLevel'].value_counts()

Out[55]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [56]: df['BPLevel'].value_counts()

Out[56]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [57]: df['BPLevel'].value_counts()

Out[57]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [58]: df['BPLevel'].value_counts()

Out[58]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [59]: df['BPLevel'].value_counts()

Out[59]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [60]: df['BPLevel'].value_counts()

Out[60]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [61]: df['BPLevel'].value_counts()

Out[61]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [62]: df['BPLevel'].value_counts()

Out[62]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [63]: df['BPLevel'].value_counts()

Out[63]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [64]: df['BPLevel'].value_counts()

Out[64]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [65]: df['BPLevel'].value_counts()

Out[65]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [66]: df['BPLevel'].value_counts()

Out[66]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [67]: df['BPLevel'].value_counts()

Out[67]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [68]: df['BPLevel'].value_counts()

Out[68]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [69]: df['BPLevel'].value_counts()

Out[69]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [70]: df['BPLevel'].value_counts()

Out[70]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [71]: df['BPLevel'].value_counts()

Out[71]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [72]: df['BPLevel'].value_counts()

Out[72]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [73]: df['BPLevel'].value_counts()

Out[73]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [74]: df['BPLevel'].value_counts()

Out[74]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [75]: df['BPLevel'].value_counts()

Out[75]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [76]: df['BPLevel'].value_counts()

Out[76]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [77]: df['BPLevel'].value_counts()

Out[77]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [78]: df['BPLevel'].value_counts()

Out[78]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [79]: df['BPLevel'].value_counts()

Out[79]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [80]: df['BPLevel'].value_counts()

Out[80]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [81]: df['BPLevel'].value_counts()

Out[81]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [82]: df['BPLevel'].value_counts()

Out[82]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [83]: df['BPLevel'].value_counts()

Out[83]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [84]: df['BPLevel'].value_counts()

Out[84]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [85]: df['BPLevel'].value_counts()

Out[85]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [86]: df['BPLevel'].value_counts()

Out[86]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [87]: df['BPLevel'].value_counts()

Out[87]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [88]: df['BPLevel'].value_counts()

Out[88]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [89]: df['BPLevel'].value_counts()

Out[89]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [90]: df['BPLevel'].value_counts()

Out[90]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [91]: df['BPLevel'].value_counts()

Out[91]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [92]: df['BPLevel'].value_counts()

Out[92]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [93]: df['BPLevel'].value_counts()

Out[93]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [94]: df['BPLevel'].value_counts()

Out[94]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [95]: df['BPLevel'].value_counts()

Out[95]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [96]: df['BPLevel'].value_counts()

Out[96]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [97]: df['BPLevel'].value_counts()

Out[97]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [98]: df['BPLevel'].value_counts()

Out[98]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [99]: df['BPLevel'].value_counts()

Out[99]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64

In [100]: df['BPLevel'].value_counts()

Out[100]:
normal    787
high      211
low        25
High        5
Low         3
normal      1
Name: BPLevel, dtype: int64
```

Figure-4 Printing sample data

```
Age
less than 40    488
40-49           164
50-59           156
60 or older     144
Name: Age, dtype: int64

Gender
Male           580
Female         372
Name: Gender, dtype: int64

Family_Diabetes
no             498
yes            454
Name: Family_Diabetes, dtype: int64

highBP
no             724
yes            228
Name: highBP, dtype: int64

PhysicallyActive
less than half an hr    336
more than half an hr   272
one hr or more          212
none                    132
Name: PhysicallyActive, dtype: int64

Smoking
no             844
yes            188
Name: Smoking, dtype: int64

Alcohol
no             788
yes            192
Name: Alcohol, dtype: int64

RegularMedicine
no             815
yes            336
o              1
Name: RegularMedicine, dtype: int64

JunkFood
occasionally    672
often           184
very often      52
always          44
Name: JunkFood, dtype: int64

Stress
sometimes       564
very often      164
not at all      136
always          88
Name: Stress, dtype: int64

BPLevel
normal          788
high            211
low              25
High             5
Low              3
Name: BPLevel, dtype: int64

Pdiabetes
0              936
yes            34
no              1
Name: Pdiabetes, dtype: int64

UrinationFreq
not much        864
quite often     288
Name: UrinationFreq, dtype: int64

Diabetic
no             885
yes            386
Name: Diabetic, dtype: int64
```

Figure-5 Printing the column with binary Values

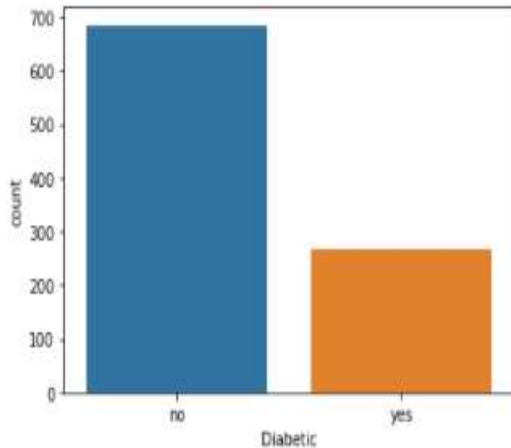


Figure-6 Diabetic data graph

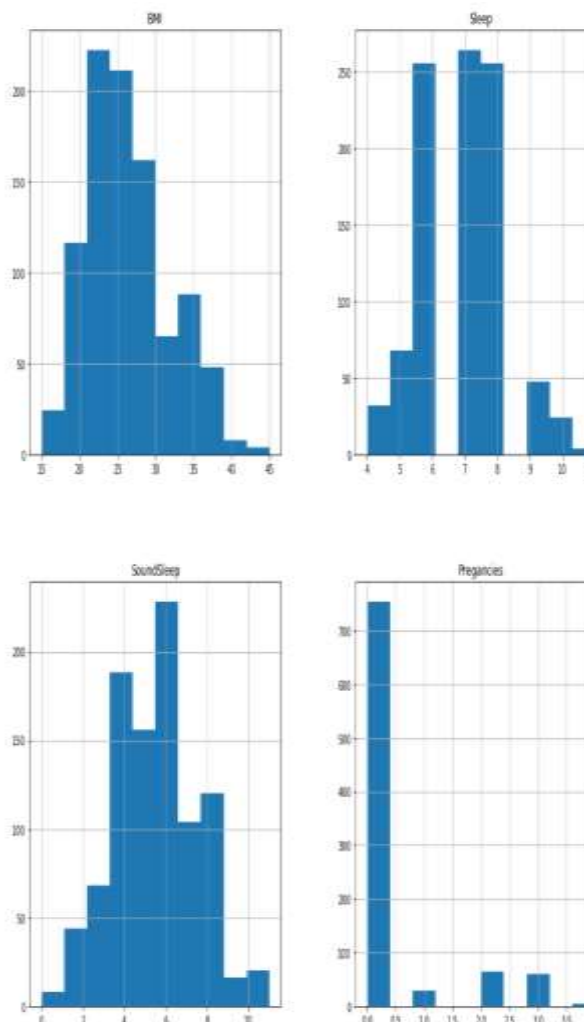


Figure-7 concatenating the all variables into one dataframe

```
In [19]: knn_clf = KNeighborsClassifier()
knn_clf.fit(X_train, y_train)
y_pred = knn_clf.predict(X_test)
print(classification_report(y_test, y_pred))

precision    recall  f1-score   support

0           0.84      0.84      0.84        58
1           0.85      0.85      0.85        27

accuracy:      0.86      0.86      0.86        86
macro avg:      0.85      0.85      0.85        86
weighted avg:  0.85      0.85      0.85        86

In [20]: rf_clf = RandomForestClassifier(n_estimators=500, max_leaf_nodes=16, n_jobs=-1)
rf_clf.fit(X_train, y_train)
y_pred_rf = rf_clf.predict(X_test)
print(classification_report(y_test, y_pred_rf))

precision    recall  f1-score   support

0           0.86      0.84      0.85        58
1           0.83      0.74      0.78        27

accuracy:      0.87      0.84      0.85        86
macro avg:      0.85      0.80      0.83        86
weighted avg:  0.85      0.80      0.83        86

In [21]: param_grid = {
    'n_estimators': [100, 400, 500, 800],
    'max_features': [10, 12, 14, 16, 18, 20, 22, 24, 26, 28],
    'bootstrap': [False],
    'n_jobs': [100, 200, 300, 400, 500, 600],
    'max_depth': [2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30]
}

grid_search = GridSearchCV(rf_clf, param_grid, cv=5, n_jobs=-1)
grid_search.fit(X_train, y_train)
```

Figure-8 Prediction using algorithms

#### IV. CONCLUSION

evaluation from the following results suggests that the risk of growing diabetes is high in folks that eat a number of junk meals. according to a current look at, someone inside the metropolis eats much less junk food, has greater pressure, and is less active. This makes town people more likely to have diabetes than in rural areas. This prognosticator suggests the best outcomes of the modifications that want to be made to live healthful and diagnose and deal with diabetics within the early ranges. daily intake of junk food will result in diabetes mellitus to save you this with the aid of lowering the consumption of junk meals and the use of greater organic meals. the main use of research to enhance the fee of drugs for diabetes mellitus. This records can be used to expect any future infection. This observe is currently learning and enhancing system studying techniques for predicting diabetes and some other ailment. The research performed here is economically possible with improved facts accuracy. Random forest works a whole lot higher than the ok-NN categories. in the end, it's miles believed that those findings will be beneficial for researchers running on independent diabetes prognosis.

## V. REFERENCES

1. Prediction of Type 2 Diabetes using Machine Learning Classification Methods -Neha Prerna Tigga  
<https://www.sciencedirect.com/science/article/pii/S1877050920308024>
2. Urban-Rural Differences in the Prevalence of Self-Reported Diabetes and its Risk Factors-Zahra Khorrami, MS, Shahin Yarahmadi  
<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5722966/>
3. Machine learning and artificial intelligence based Diabetes Mellitus detection- S.Ananda Theertan ,S.Thillai Ganesh, S.KCidham  
<https://www.sciencedirect.com/science/article/pii/S1319157820304134>
4. Kaggle Dataset- Neha Prerna Tigga and Dr. Shruti Garg <https://www.kaggle.com/hadiaskari1981/knn-classifier/data>
5. Introduction to K-Nearest Neighbours: A Powerful Machine Learning Algorithm- Tavish Srivastava  
<https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/>