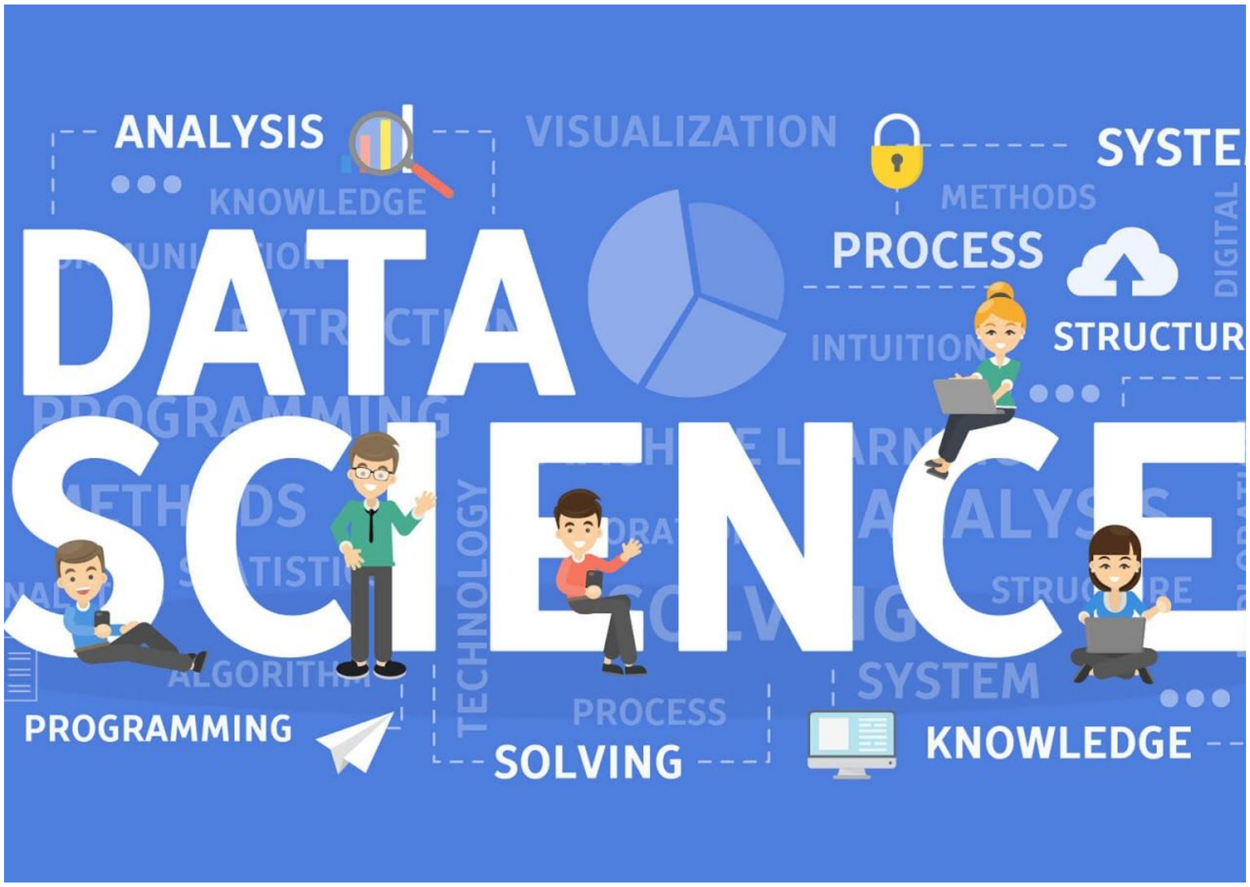
IDS REPORT

ML CLASSIFICATION



Group members

Tushar Singhal (18ucs013)

Pooja Singhal (18ucs072)

Pulkit Jain (18ucs163)

Ayush Rathi (18ucs227)

Index

Project goal

The goal of this Machine Learning project is to visualize the symptoms and factors on which the positive or negative cases of diabetes depend and also the relationships among themselves. This project also aims at predicting whether a person is diabetic or not based on symptoms, age, gender using Gaussian naive bayes and Decision tree classifier.

Introduction To Dataset

The Diabetes database was collected by us from the UCI Donald Bren School Of Information and Computer archives who confirm that this has been collected using direct questionnaires from the patients of Sylhet Diabetes Hospital in Sylhet, Bangladesh and approved by a doctor.

Source of this dataset:

https://archive.ics.uci.edu/ml/machine-learning-databases/00529/

The dataset includes various information regarding diabetic and non-diabetic people. It lists various symptoms associated with diabetes and also tells us about the age and gender of the person.

The Information makes it possible for the people within a certain age group and gender to change their dietary plans and other lifestyle changes by predicting the risk of diabetes at an early stage.

Column Description:

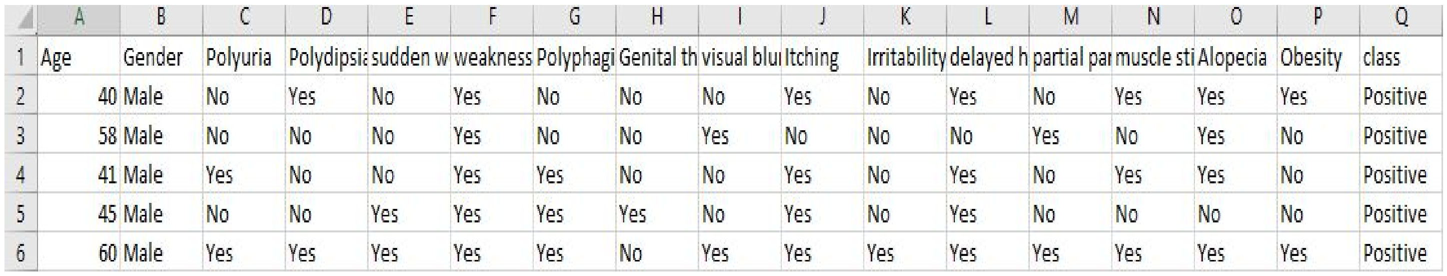
No of Columns/attributes/features = **17**

Column	Type (Range of values)
Age	Numerical (16 to 90)
Gender	Categorical (Male, Female)
Polyuria	Categorical (Yes, No)
Polydipsia	Categorical (Yes, No)
Sudden weight loss	Categorical (Yes, No)
Weakness	Categorical (Yes, No)
Polyphagia	Categorical (Yes, No)
Genital thrush	Categorical (Yes, No)
Visual blurring	Categorical (Yes, No)
Itching	Categorical (Yes, No)
Irritability	Categorical (Yes, No)
Delayed healing	Categorical (Yes, No)
Partial paresis	Categorical (Yes, No)
Muscle stiffness	Categorical (Yes, No)
Alopecia	Categorical (Yes, No)
Obesity	Categorical (Yes, No)
Class	Categorical (Positive, Negative)

Record description:

No of records = **520**

Sample records



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	Age	Gender	Polyuria	Polydipsia	sudden w	weakness	Polyphagi	Genital th	visual blu	Itching	Irritability	delayed h	partial par	muscle sti	Alopecia	Obesity	class
2	40	Male	No	Yes	No	Yes	No	No	No	Yes	No	Yes	No	Yes	Yes	Yes	Positive
3	58	Male	No	No	No	Yes	No	No	Yes	No	No	No	Yes	No	Yes	No	Positive
4	41	Male	Yes	No	No	Yes	Yes	No	No	Yes	No	Yes	No	Yes	Yes	No	Positive
5	45	Male	No	No	Yes	Yes	Yes	Yes	No	Yes	No	Yes	No	No	No	No	Positive
6	60	Male	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Positive

Characteristics of the dataset:

1) Multivariate

2) Area: Medical

3) Attribute Characteristics: Categorical, Integer

4) Associated Tasks: Classification

5) Missing Values: No

6) No of attributes = 17

7) No of records = 520

Steps to classify the dataset

1.) Importing libraries and dataset to be used to classify and preprocess the dataset Here we would need numpy for the mathematical calculation of the working with dataset and matrices. Then we would need pandas which have many inbuilt functions to help performing various kinds of slices and selections. Matplotlib and seaborn will be used for visualizations and sklearn has important libraries related to applying algorithms.

2.) Preprocessing the dataset will include checking for missing values, handling outliers, splitting the dataset into training and test dataset.

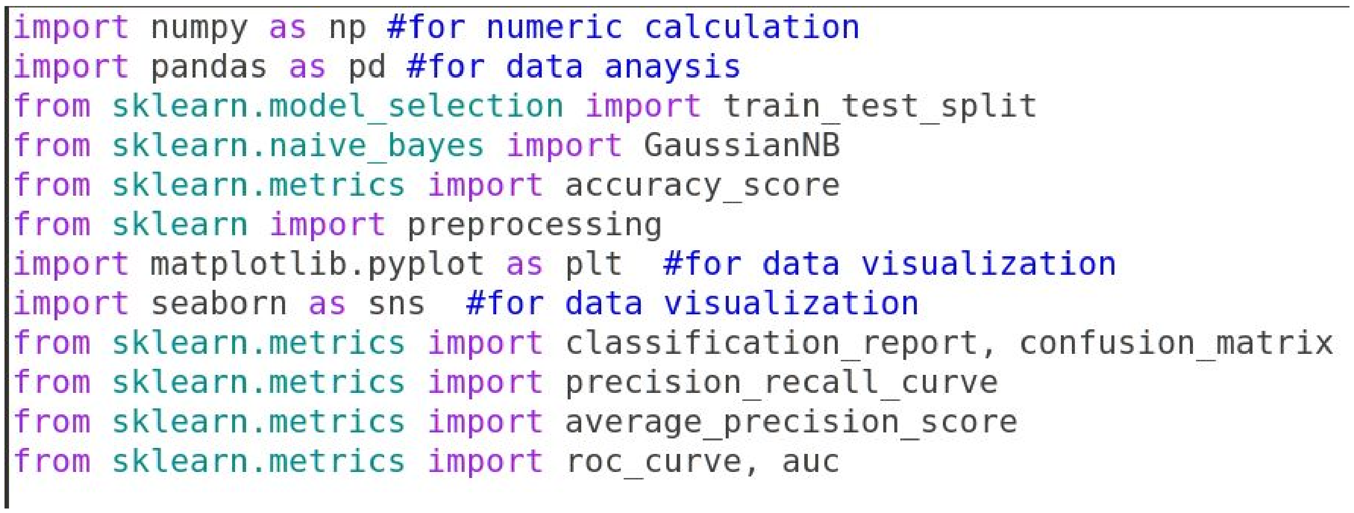
3.) The categorical data needs to be converted to integer type to classify the dataset using label encoder from the preprocessing library.

4.) After completing the preprocessing we analyze the dataset to gain inferences and use matplot and seaborn library to visualize the data.

5.) We choose the algorithms to be used for classification as per our dataset and apply them using the inbuilt libraries of python.

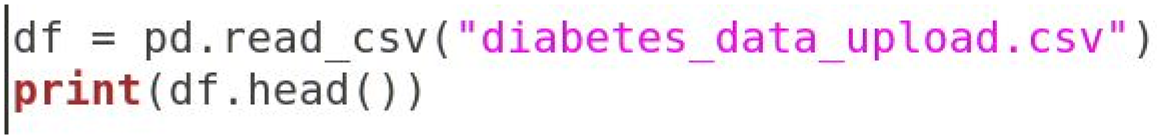
6.) We analyze our classification algorithm using the precision, recall, accuracy and f1 score of the model and compare the different algorithms used according to their performance.

Importing Libraries

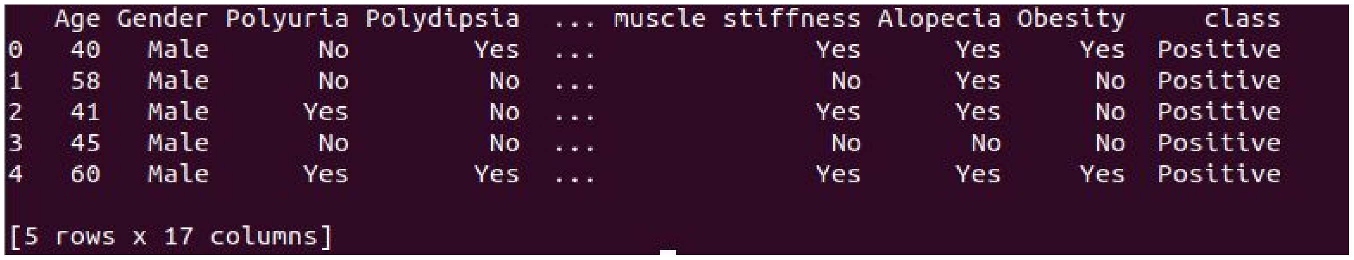


```
import numpy as np #for numeric calculation
import pandas as pd #for data anaysis
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
from sklearn import preprocessing
import matplotlib.pyplot as plt  #for data visualization
import seaborn as sns  #for data visualization
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import average_precision_score
from sklearn.metrics import roc_curve, auc
```

Importing Dataset



```
df = pd.read_csv("diabetes_data_upload.csv")
print(df.head())
```



```
   Age Gender Polyuria Polydipsia  ... muscle stiffness Alopecia Obesity     class
0   40   Male       No        Yes  ...              Yes      Yes     Yes  Positive
1   58   Male       No         No  ...               No      Yes      No  Positive
2   41   Male      Yes         No  ...              Yes      Yes      No  Positive
3   45   Male       No         No  ...               No       No      No  Positive
4   60   Male      Yes        Yes  ...              Yes      Yes     Yes  Positive

[5 rows x 17 columns]
```

Data Preprocessing

Handling Missing Values: We had no missing values in our dataset so handling the missing values part was not required for our dataset.

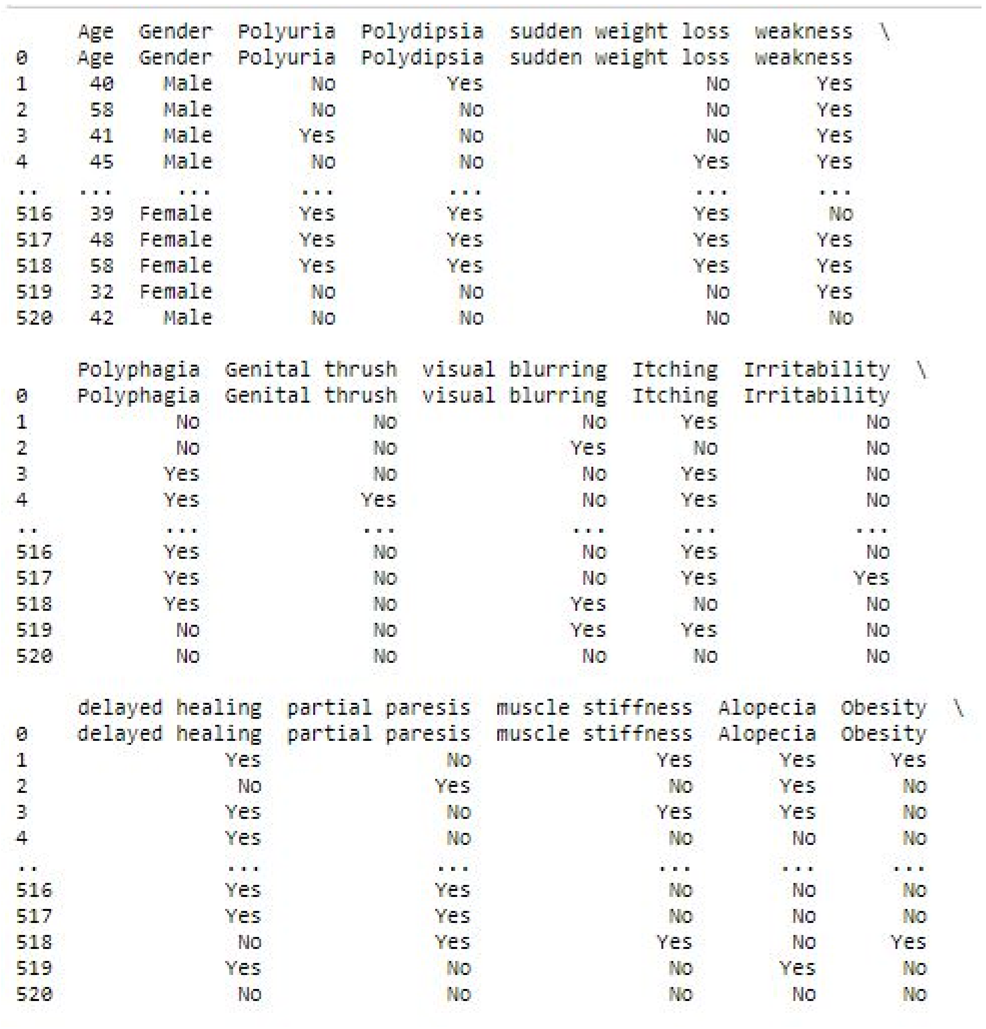
Handling Outliers: All our attributes except age were categorical and took values as either ‘yes’ or ‘no’, and the class labels took values as either ‘positive’, or ‘negative’ so there were no outliers in the data that need to be preprocessed explicitly.

Transforming Data: The only part that required preprocessing was converting the categorical attributes that had string values into integer values because the inbuilt classifier requires the data to be preprocessed and transformed into integer values before the model can be trained and classified.

We have made use of the fit_transform() function of label encoder, present in the preprocessing library of sklearn, this function preprocesses the data passed as a parameter and converts it into integer type data ready to be processed, the original data can be retrieved back with the help of inverse_transform() function of label encoder

.

Data before transformation



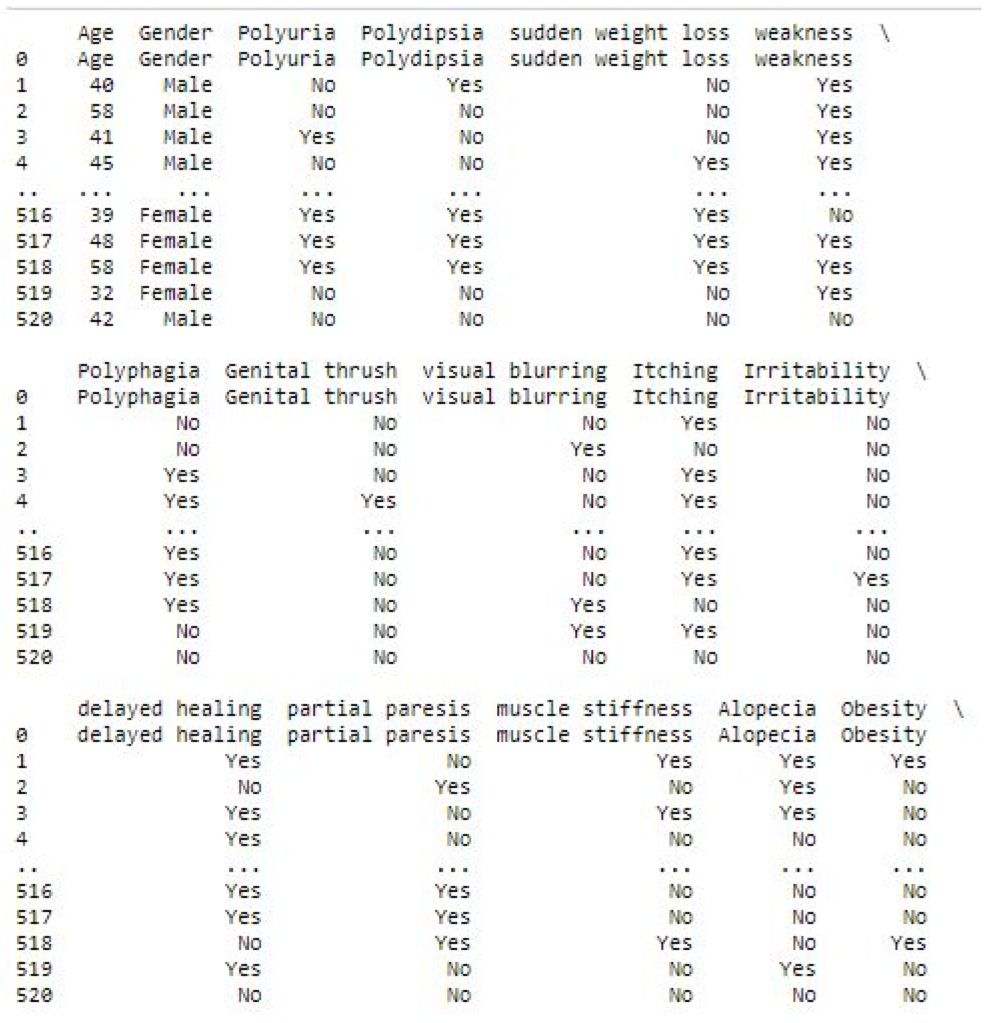
```
     Age  Gender  Polyuria  Polydipsia  sudden weight loss  weakness  \
0    Age  Gender  Polyuria  Polydipsia  sudden weight loss  weakness
1     40    Male        No         Yes                  No       Yes
2     58    Male        No          No                  No       Yes
3     41    Male       Yes          No                  No       Yes
4     45    Male        No          No                 Yes       Yes
..   ...     ...       ...         ...                 ...       ...
516   39  Female       Yes         Yes                 Yes        No
517   48  Female       Yes         Yes                 Yes       Yes
518   58  Female       Yes         Yes                 Yes       Yes
519   32  Female        No          No                  No       Yes
520   42    Male        No          No                  No        No

     Polyphagia  Genital thrush  visual blurring  Itching  Irritability  \
0    Polyphagia  Genital thrush  visual blurring  Itching  Irritability
1            No              No               No      Yes            No
2            No              No              Yes       No            No
3           Yes              No               No      Yes            No
4           Yes             Yes               No      Yes            No
..          ...             ...              ...      ...           ...
516         Yes              No               No      Yes            No
517         Yes              No               No      Yes           Yes
518         Yes              No              Yes       No            No
519          No              No              Yes      Yes            No
520          No              No               No       No            No

     delayed healing  partial paresis  muscle stiffness  Alopecia  Obesity  \
0    delayed healing  partial paresis  muscle stiffness  Alopecia  Obesity
1                Yes               No               Yes       Yes      Yes
2                 No              Yes                No       Yes       No
3                Yes               No               Yes       Yes       No
4                Yes               No                No        No       No
..               ...              ...               ...       ...      ...
516              Yes              Yes                No        No       No
517              Yes              Yes                No        No       No
518               No              Yes               Yes        No      Yes
519              Yes               No                No       Yes       No
520               No               No                No        No       No
```

```
In [64]: from sklearn import preprocessing
In [65]: le = preprocessing.LabelEncoder()
         data['Age'] = le.fit_transform(data['Age'])
         data['Gender'] = le.fit_transform(data['Gender'])
         data['Polyuria'] = le.fit_transform(data['Polyuria'])
         data['Polydipsia'] = le.fit_transform(data['Polydipsia'])
         data['sudden weight loss'] = le.fit_transform(data['sudden weight loss'])
         data['weakness'] = le.fit_transform(data['weakness'])
         data['Polyphagia'] = le.fit_transform(data['Polyphagia'])
         data['Genital thrush'] = le.fit_transform(data['Genital thrush'])
         data['visual blurring'] = le.fit_transform(data['visual blurring'])
         data['Itching'] = le.fit_transform(data['Itching'])
         data['Irritability'] = le.fit_transform(data['Irritability'])
         data['delayed healing'] = le.fit_transform(data['delayed healing'])
         data['partial paresis'] = le.fit_transform(data['partial paresis'])
         data['muscle stiffness'] = le.fit_transform(data['muscle stiffness'])
         data['Alopecia'] = le.fit_transform(data['Alopecia'])
         data['Obesity'] = le.fit_transform(data['Obesity'])
```

```
In [66]: X = data[attributes]
```

```
In [67]: print(X)
```

Data after transformation



```
     Age  Gender  Polyuria  Polydipsia  sudden weight loss  weakness  \
0    Age  Gender  Polyuria  Polydipsia  sudden weight loss  weakness
1     40    Male        No         Yes                  No       Yes
2     58    Male        No          No                  No       Yes
3     41    Male       Yes          No                  No       Yes
4     45    Male        No          No                 Yes       Yes
..   ...     ...       ...         ...                 ...       ...
516   39  Female       Yes         Yes                 Yes        No
517   48  Female       Yes         Yes                 Yes       Yes
518   58  Female       Yes         Yes                 Yes       Yes
519   32  Female        No          No                  No       Yes
520   42    Male        No          No                  No        No

     Polyphagia  Genital thrush  visual blurring  Itching  Irritability  \
0    Polyphagia  Genital thrush  visual blurring  Itching  Irritability
1            No              No               No      Yes            No
2            No              No              Yes       No            No
3           Yes              No               No      Yes            No
4           Yes             Yes               No      Yes            No
..          ...             ...              ...      ...           ...
516         Yes              No               No      Yes            No
517         Yes              No               No      Yes           Yes
518         Yes              No              Yes       No            No
519          No              No              Yes      Yes            No
520          No              No               No       No            No

     delayed healing  partial paresis  muscle stiffness  Alopecia  Obesity  \
0    delayed healing  partial paresis  muscle stiffness  Alopecia  Obesity
1                Yes               No               Yes       Yes      Yes
2                 No              Yes                No       Yes       No
3                Yes               No               Yes       Yes       No
4                Yes               No                No        No       No
..               ...              ...               ...       ...      ...
516              Yes              Yes                No        No       No
517              Yes              Yes                No        No       No
518               No              Yes               Yes        No      Yes
519              Yes               No                No       Yes       No
520               No               No                No        No       No
```

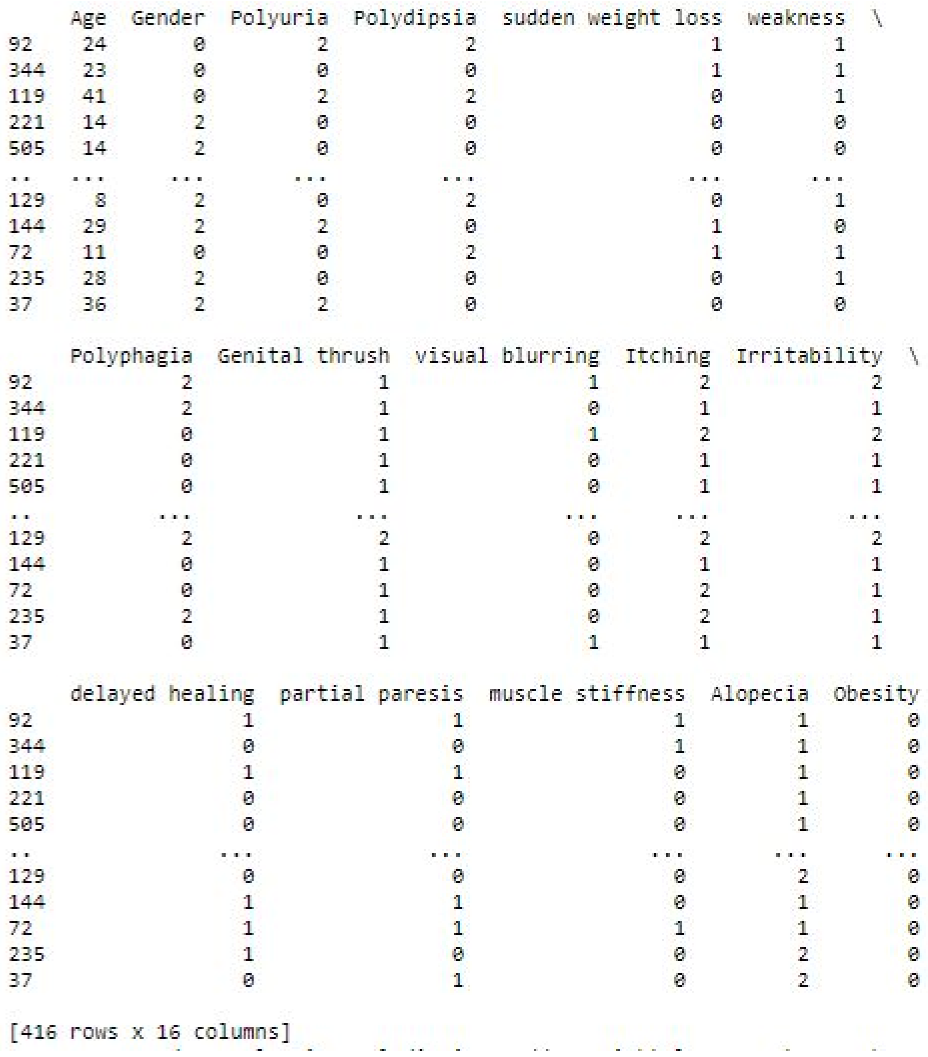
Splitting the Data: After transforming the data, and making it ready to be trained and tested, we need to split the data into a test and train data set. Here, the train set will be used to train the model by fitting the data into the classifier and the test data will be used to make predictions using the trained classifier and thus predict the accuracy of the model. We have made 80-20 split i.e. 80% of the available data will be used in training of the classifier model and the rest 20% of the data will be used to test the trained model.

We have used the train_test_split() function of the sklearn.model_selection() library to split the dataset into test and train datasets.

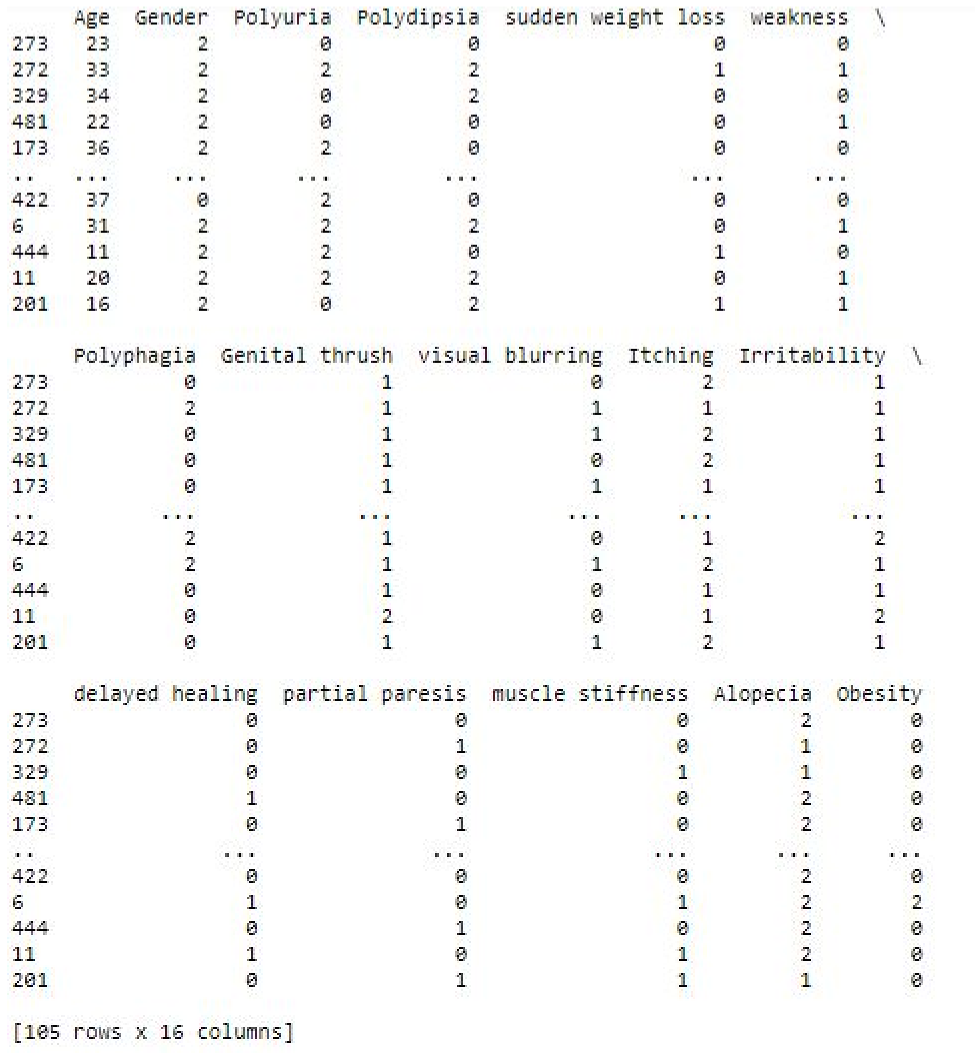
```
In [68]: from sklearn.model_selection import train_test_split
```

```
In [69]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
         random_state=1)
```

Train and Test Dataset



```
     Age  Gender  Polyuria  Polydipsia  sudden weight loss  weakness  \
92    24       0         2           2                   1         1
344   23       0         0           0                   1         1
119   41       0         2           2                   0         1
221   14       2         0           0                   0         0
505   14       2         0           0                   0         0
..   ...     ...       ...         ...                 ...       ...
129    8       2         0           2                   0         1
144   29       2         2           0                   1         0
72    11       0         0           2                   1         1
235   28       2         0           0                   0         1
37    36       2         2           0                   0         0

     Polyphagia  Genital thrush  visual blurring  Itching  Irritability  \
92            2               1                1        2             2
344           2               1                0        1             1
119           0               1                1        2             2
221           0               1                0        1             1
505           0               1                0        1             1
..          ...             ...              ...      ...           ...
129           2               2                0        2             2
144           0               1                0        1             1
72            0               1                0        2             1
235           2               1                0        2             1
37            0               1                1        1             1

     delayed healing  partial paresis  muscle stiffness  Alopecia  Obesity
92                 1                1                 1         1        0
344                0                0                 1         1        0
119                1                1                 0         1        0
221                0                0                 0         1        0
505                0                0                 0         1        0
..               ...              ...               ...       ...      ...
129                0                0                 0         2        0
144                1                1                 0         1        0
72                 1                1                 1         1        0
235                1                0                 0         2        0
37                 0                1                 0         2        0

[416 rows x 16 columns]
```

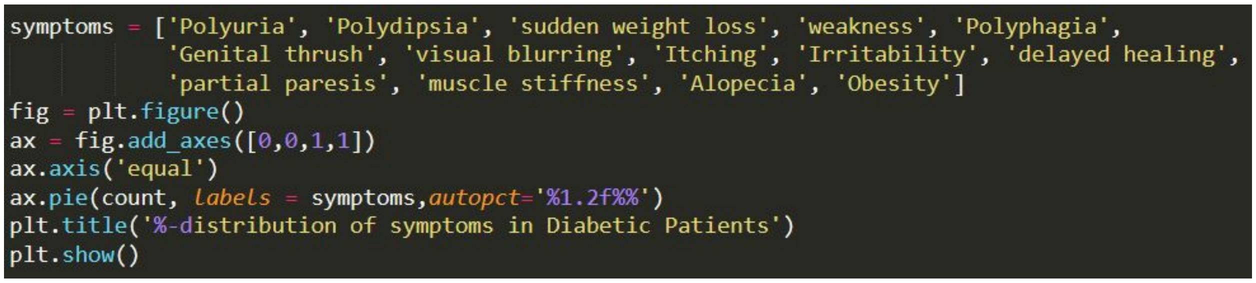


```
     Age  Gender  Polyuria  Polydipsia  sudden weight loss  weakness  \
273   23       2         0           0                   0         0
272   33       2         2           2                   1         1
329   34       2         0           2                   0         0
481   22       2         0           0                   0         1
173   36       2         2           0                   0         0
..   ...     ...       ...         ...                 ...       ...
422   37       0         2           0                   0         0
6     31       2         2           2                   0         1
444   11       2         2           0                   1         0
11    20       2         2           2                   0         1
201   16       2         0           2                   1         1

     Polyphagia  Genital thrush  visual blurring  Itching  Irritability  \
273           0               1                0        2             1
272           2               1                1        1             1
329           0               1                1        2             1
481           0               1                0        2             1
173           0               1                1        1             1
..          ...             ...              ...      ...           ...
422           2               1                0        1             2
6             2               1                1        2             1
444           0               1                0        1             1
11            0               2                0        1             2
201           0               1                1        2             1

     delayed healing  partial paresis  muscle stiffness  Alopecia  Obesity
273                0                0                 0         2        0
272                0                1                 0         1        0
329                0                0                 1         1        0
481                1                0                 0         2        0
173                0                1                 0         2        0
..               ...              ...               ...       ...      ...
422                0                0                 0         2        0
6                  1                0                 1         2        2
444                0                1                 0         2        0
11                 1                0                 1         2        0
201                0                1                 1         1        0

[105 rows x 16 columns]
```

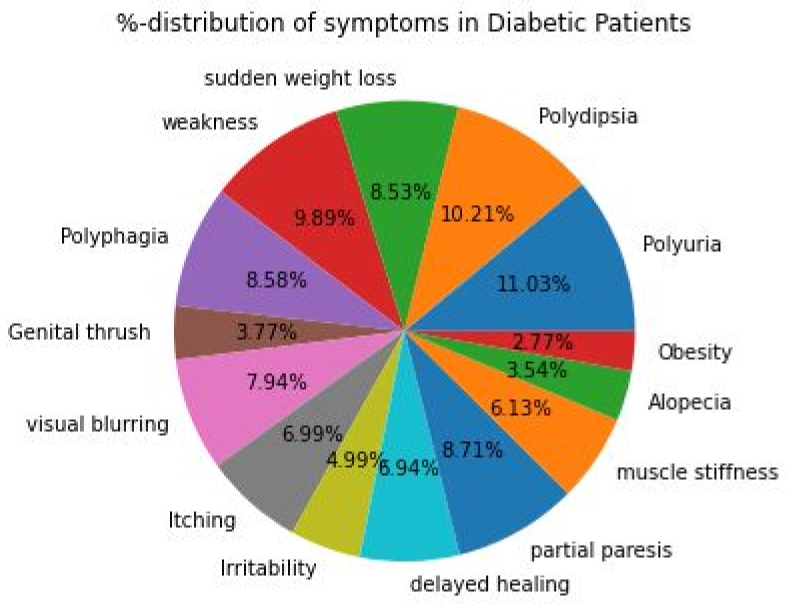
Inferences

Question thought : Which symptom is most prominent in diabetic patients ?

1. Percentage distribution of symptoms in diabetic patients



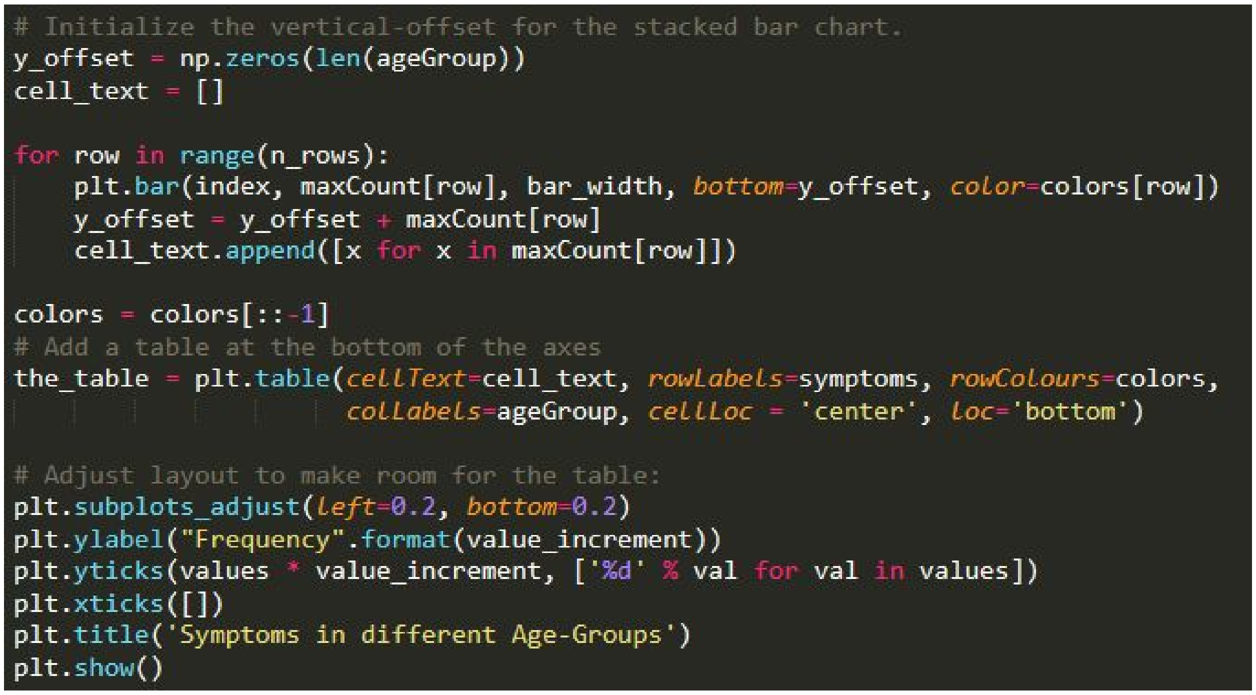
```
symptoms = ['Polyuria', 'Polydipsia', 'sudden weight loss', 'weakness', 'Polyphagia',
            'Genital thrush', 'visual blurring', 'Itching', 'Irritability', 'delayed healing',
            'partial paresis', 'muscle stiffness', 'Alopecia', 'Obesity']
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
ax.axis('equal')
ax.pie(count, labels = symptoms,autopct='%1.2f%%')
plt.title('%-distribution of symptoms in Diabetic Patients')
plt.show()
```



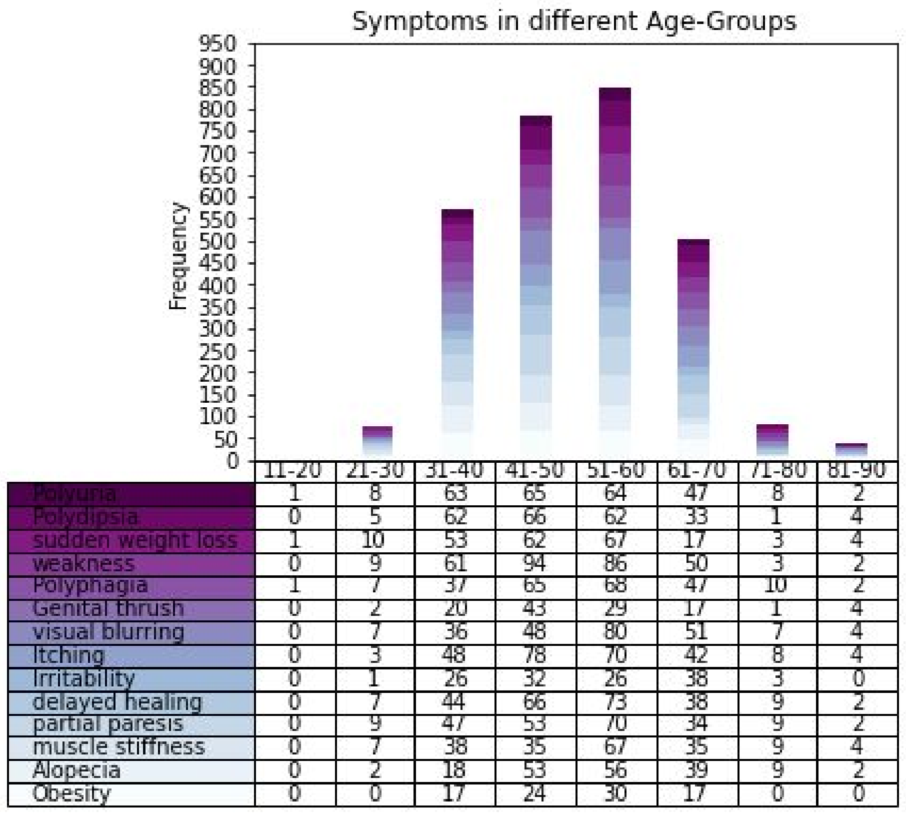
This Pie-chart shows that **Polyuria** and **Polydipsia** are symptoms which are recorded in diabitic patients. Patients showing symptoms like **Obesity** and **Alopecia** are less likely to have positive diabetes reports.

2. Frequency distribution for symptoms in different Age-Groups

- **Question Thought : Which age group is most vulnerable to diabetes symptoms ?**



```
# Initialize the vertical-offset for the stacked bar chart.
y_offset = np.zeros(len(ageGroup))
cell_text = []

for row in range(n_rows):
    plt.bar(index, maxCount[row], bar_width, bottom=y_offset, color=colors[row])
    y_offset = y_offset + maxCount[row]
    cell_text.append([x for x in maxCount[row]])

colors = colors[::-1]
# Add a table at the bottom of the axes
the_table = plt.table(cellText=cell_text, rowLabels=symptoms, rowColours=colors,
                      colLabels=ageGroup, cellLoc = 'center', loc='bottom')

# Adjust layout to make room for the table:
plt.subplots_adjust(left=0.2, bottom=0.2)
plt.ylabel("Frequency".format(value_increment))
plt.yticks(values * value_increment, ['%d' % val for val in values])
plt.xticks([])
plt.title('Symptoms in different Age-Groups')
plt.show()
```

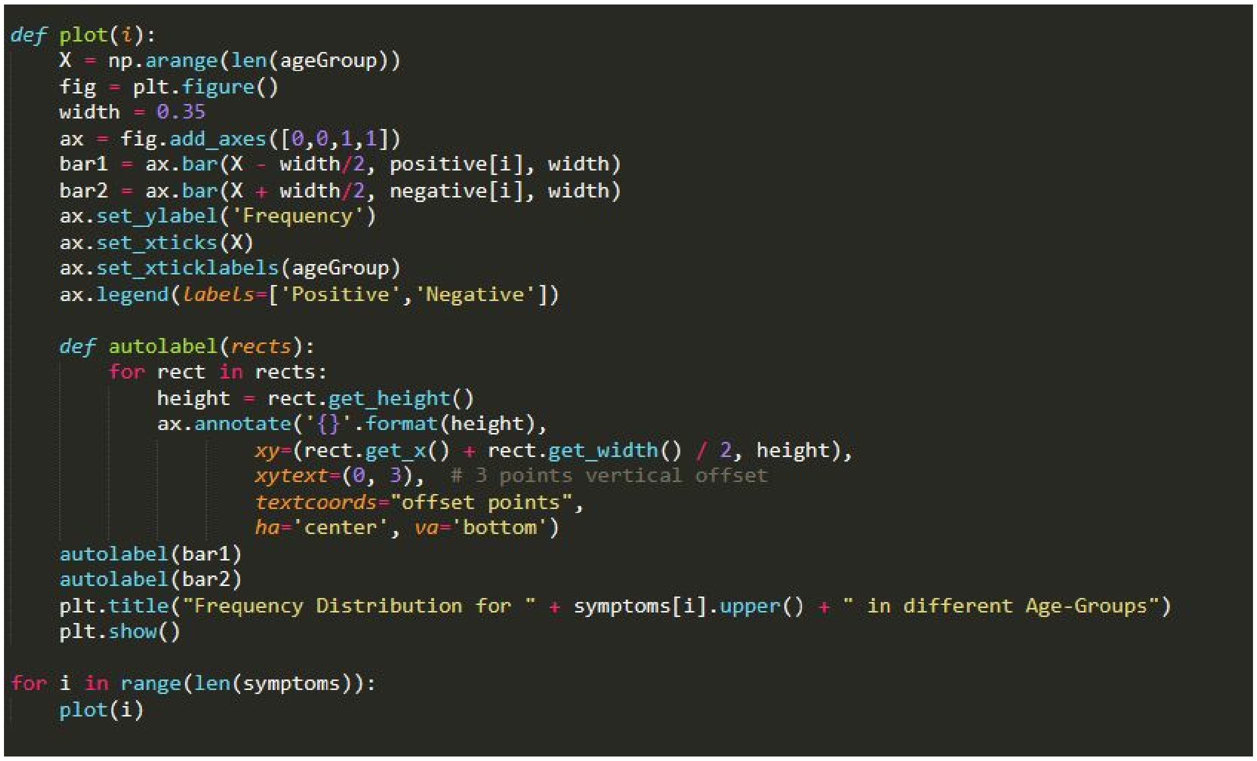


	11-20	21-30	31-40	41-50	51-60	61-70	71-80	81-90
Polyuria	1	8	63	65	64	47	8	2
Polydipsia	0	5	62	66	62	33	1	4
sudden weight loss	1	10	53	62	67	17	3	4
weakness	0	9	61	94	86	50	3	2
Polyphagia	1	7	37	65	68	47	10	2
Genital thrush	0	2	20	43	29	17	1	4
visual blurring	0	7	36	48	80	51	7	4
Itching	0	3	48	78	70	42	8	4
Irritability	0	1	26	32	26	38	3	0
delayed healing	0	7	44	66	73	38	9	2
partial paresis	0	9	47	53	70	34	9	2
muscle stiffness	0	7	38	35	67	35	9	4
Alopecia	0	2	18	53	56	39	9	2
Obesity	0	0	17	24	30	17	0	0

The above plot shows the frequency distribution of symptoms in different age groups. This is the distribution over complete dataset i.e. patients with positive as well as negative reports. It can be inferred that what symptoms are more likely to occur in which age-group. Some points that can be noted are :

- People in age-group **51-60** are the most vulnerable ones to all the symptoms.
- People in age-group 31-70 are more likely to have diabetes because all the symptoms are having a decent amount of frequency of occurrence.

3. Age-Groupwise Frequency Distribution for Symptoms Individually

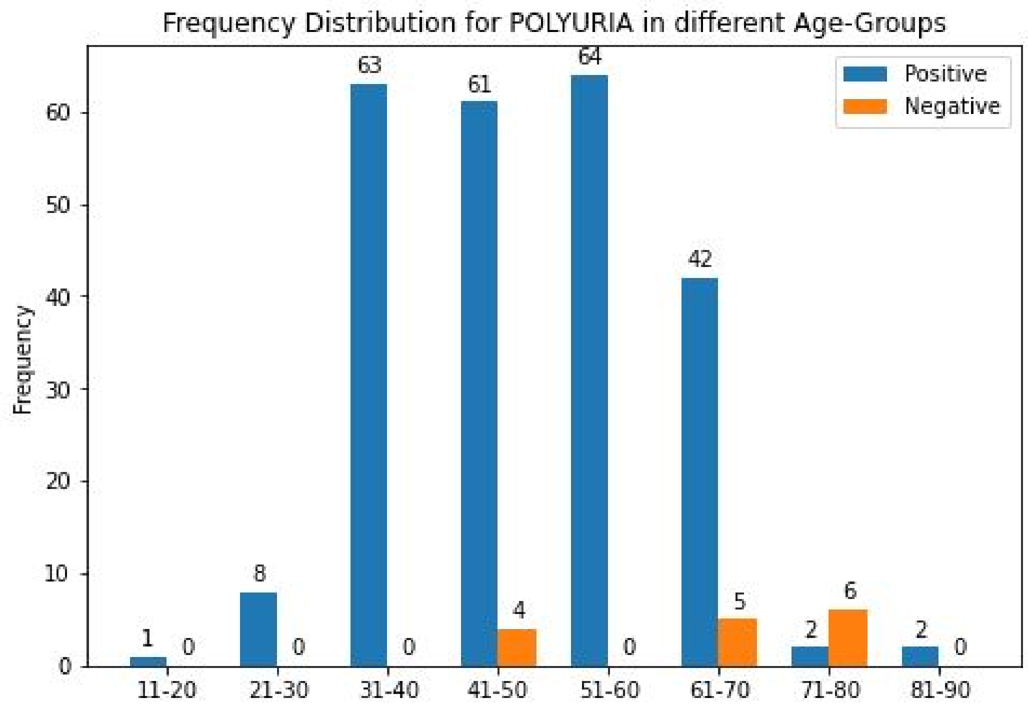


```python
def plot(i):
    X = np.arange(len(ageGroup))
    fig = plt.figure()
    width = 0.35
    ax = fig.add_axes([0,0,1,1])
    bar1 = ax.bar(X - width/2, positive[i], width)
    bar2 = ax.bar(X + width/2, negative[i], width)
    ax.set_ylabel('Frequency')
    ax.set_xticks(X)
    ax.set_xticklabels(ageGroup)
    ax.legend(labels=['Positive','Negative'])

    def autolabel(rects):
        for rect in rects:
            height = rect.get_height()
            ax.annotate('{}'.format(height),
                        xy=(rect.get_x() + rect.get_width() / 2, height),
                        xytext=(0, 3),  # 3 points vertical offset
                        textcoords="offset points",
                        ha='center', va='bottom')
    autolabel(bar1)
    autolabel(bar2)
    plt.title("Frequency Distribution for " + symptoms[i].upper() + " in different Age-Groups")
    plt.show()

for i in range(len(symptoms)):
    plot(i)
```

This segment will cover plots for individual symptoms and comparison between their occurrence in diabitic and nondiabetic patients in different age-groups.

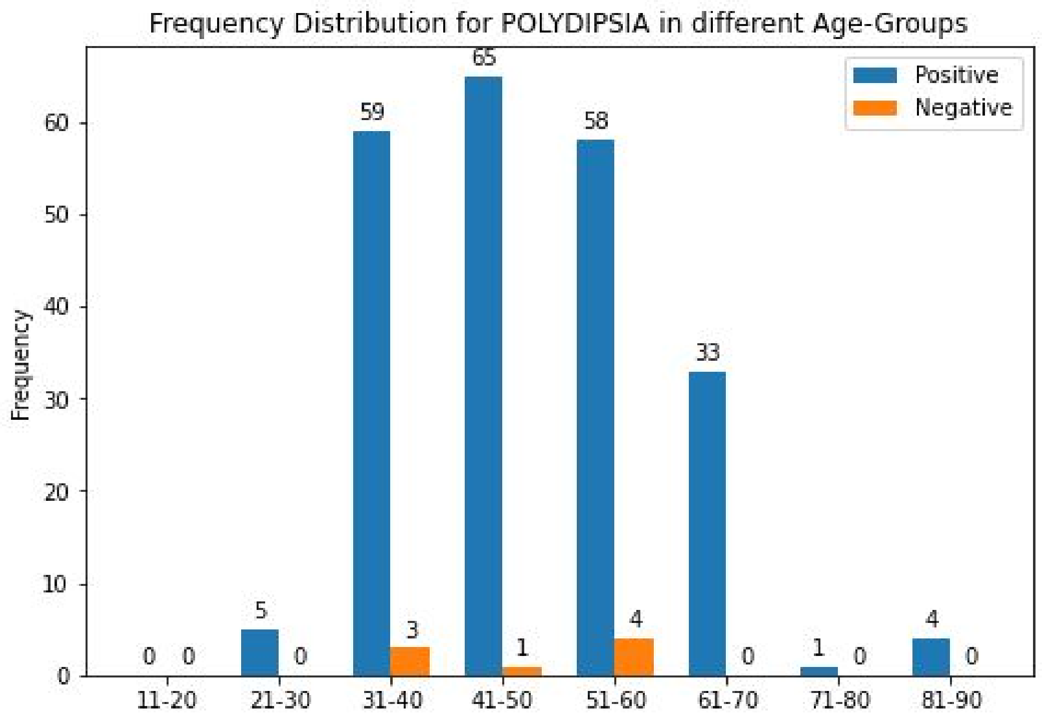
Question Thought: Which age group has the highest frequency for each symptom individually ?

3.1 Polyuria



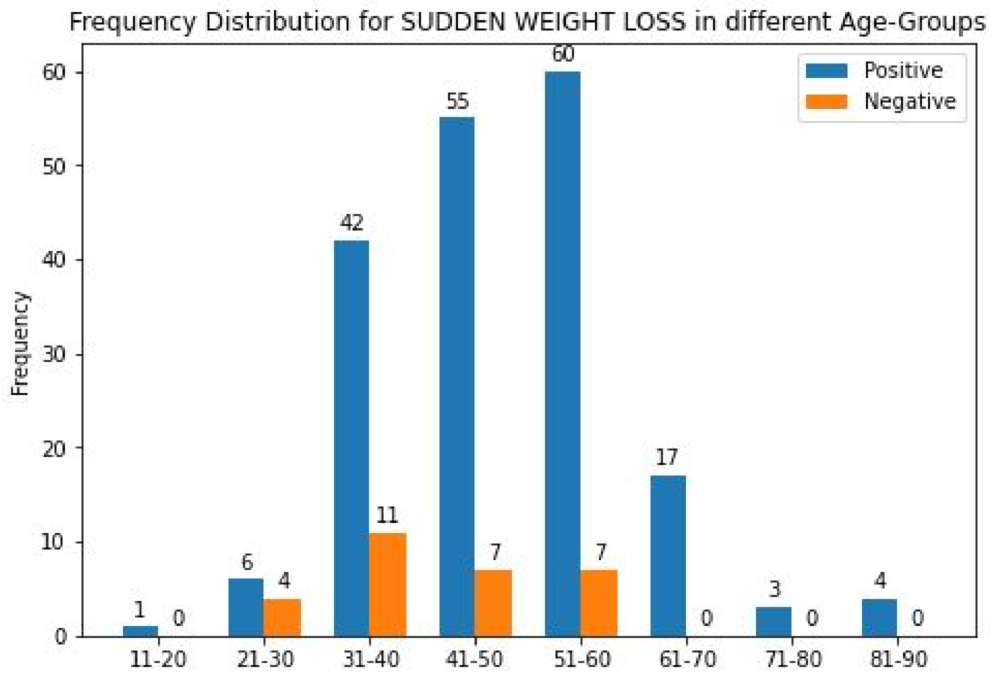
It can be inferred that people with **POLYURIA** are having high chances to have diabetes. Symptoms having high occurrence in people among the age-group **31-70**.

3.2 Polydipsia



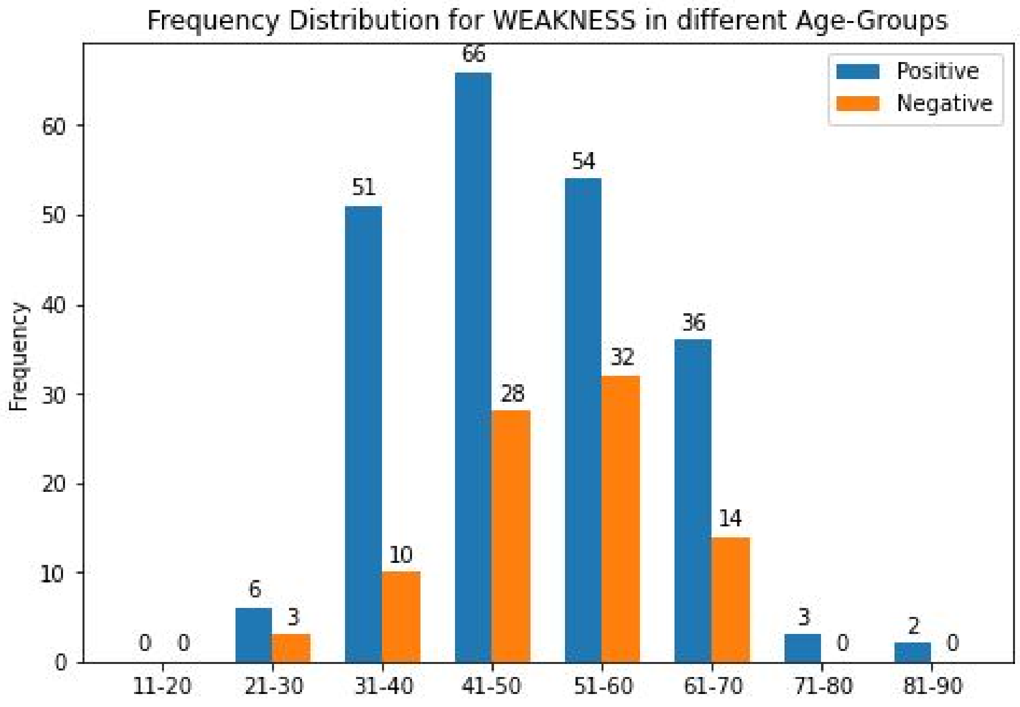
It can be inferred that people with **POLYDIPSIA** are having high chances to have diabetes. This Symptom is having high occurrence in people among the age-group **31-60**.

3.3 Sudden Weight Loss



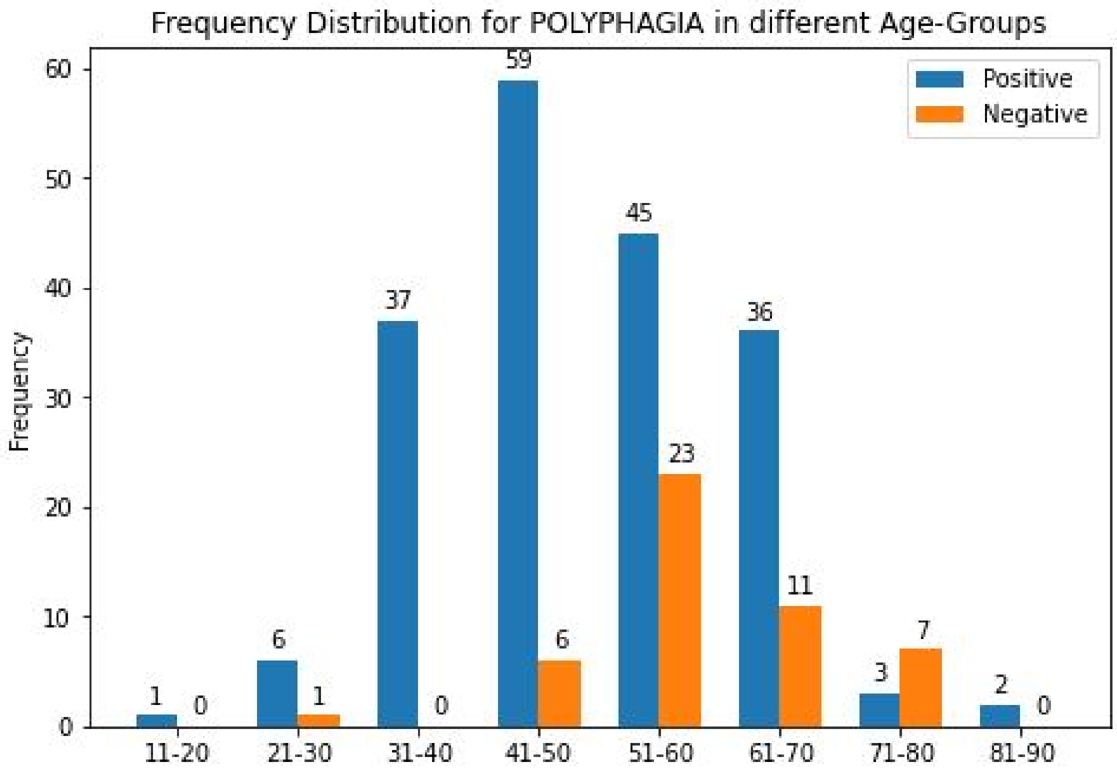
It can be inferred that people with **SUDDEN WEIGHT LOSS** are having high chances to have diabetes. This symptom has high occurrences in people among the age-group **31-60**. Also there are some cases with negative diabetes reports in spite of showing the symptom.

3.4 Weakness



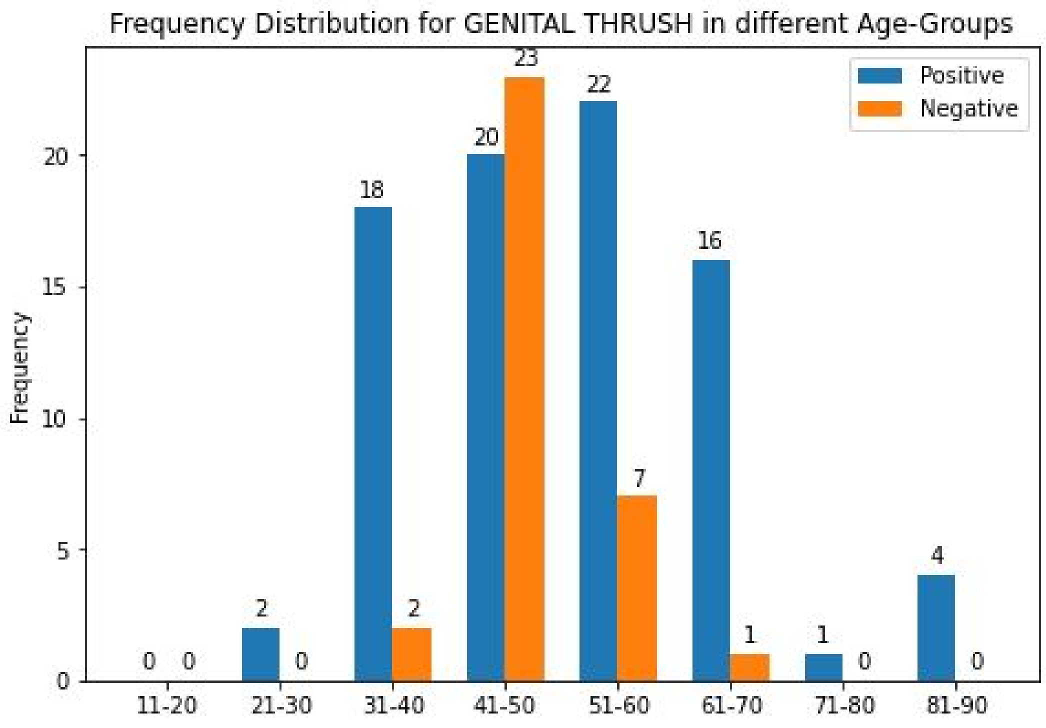
It can be inferred that people with **WEAKNESS** are having high chances to have diabetes. This Symptom is having high occurrence in people among the age-group **31-70**.There are a decent number of cases with negative diabetes reports in spite of showing the symptom.

3.5 Polyphagia



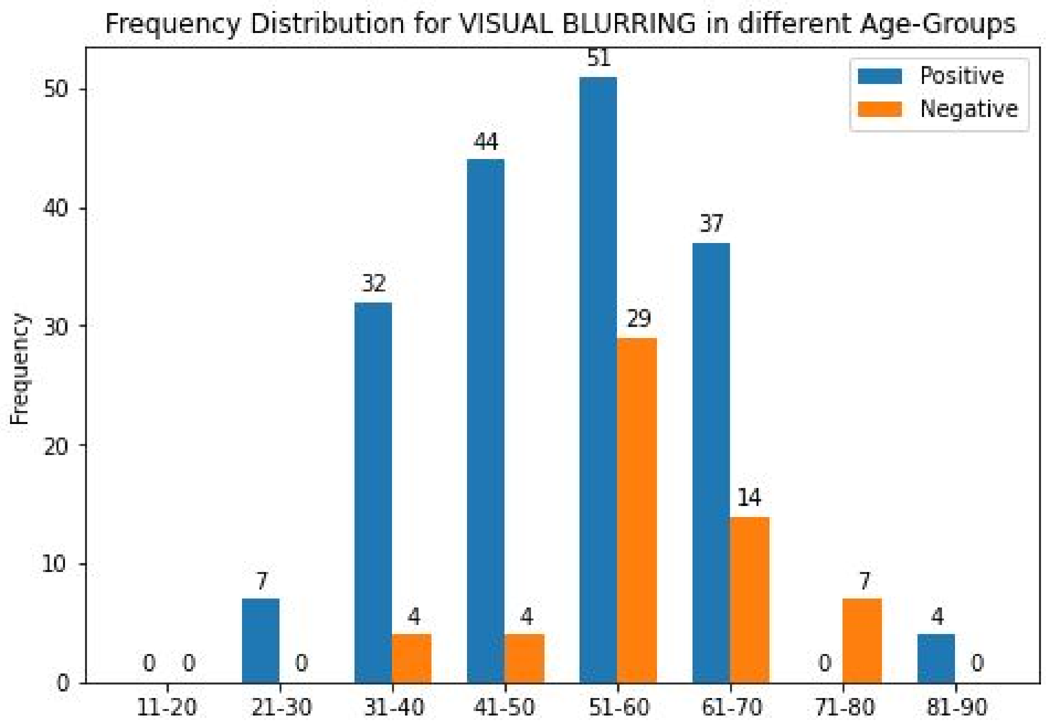
It can be inferred that people with **POLYPHAGIA** are having high chances to have diabetes. This Symptom is having high occurrence in people among the age-group **31-70**. There are some cases with negative diabetes reports in spite of showing the symptom and such cases are mainly in age-group **51-60**.

3.6 Genital Thrush



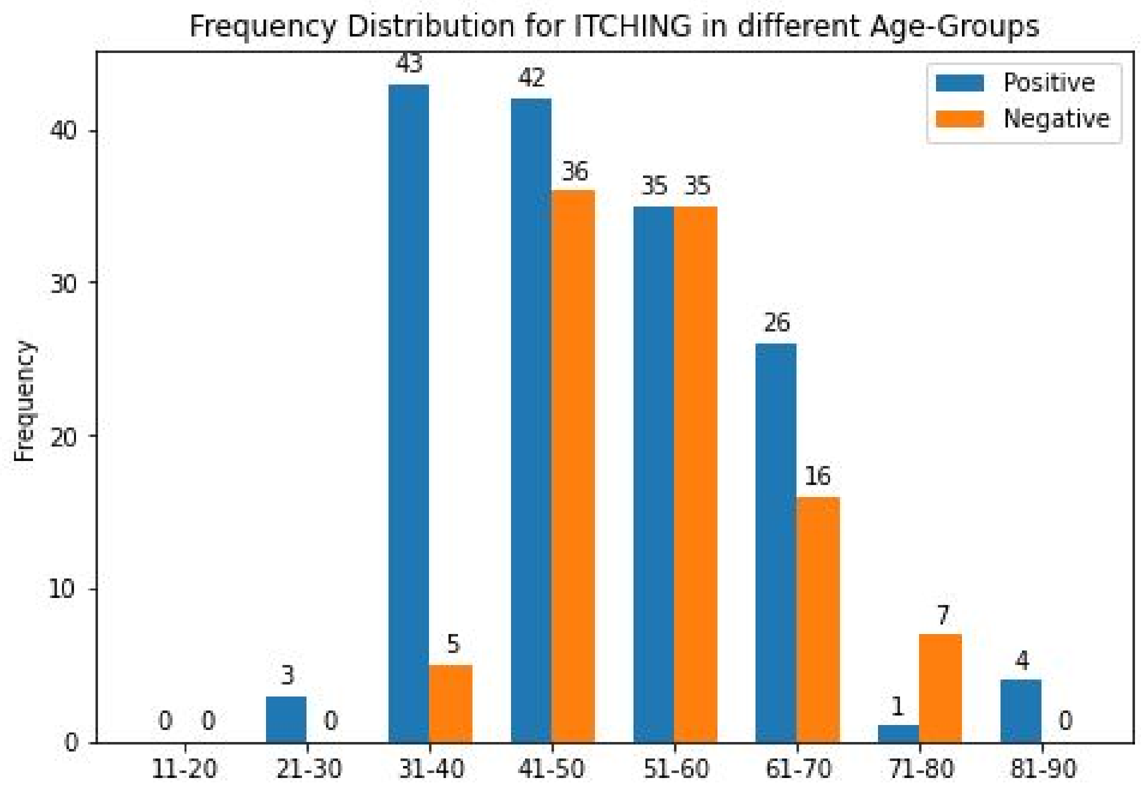
It can be inferred that people with **GENITA THRUSH** are having good chances to have diabetes. This Symptom is having good occurrence in people among the age-group **31-60**. There are a high number of cases where this symptom exists in non-diabetic patients, maily in age-group **41-50**. This is not a reliable symptom to include in the process of generating diabetes reports, especially in the above mentioned age-group.

3.7 Visual Blurring



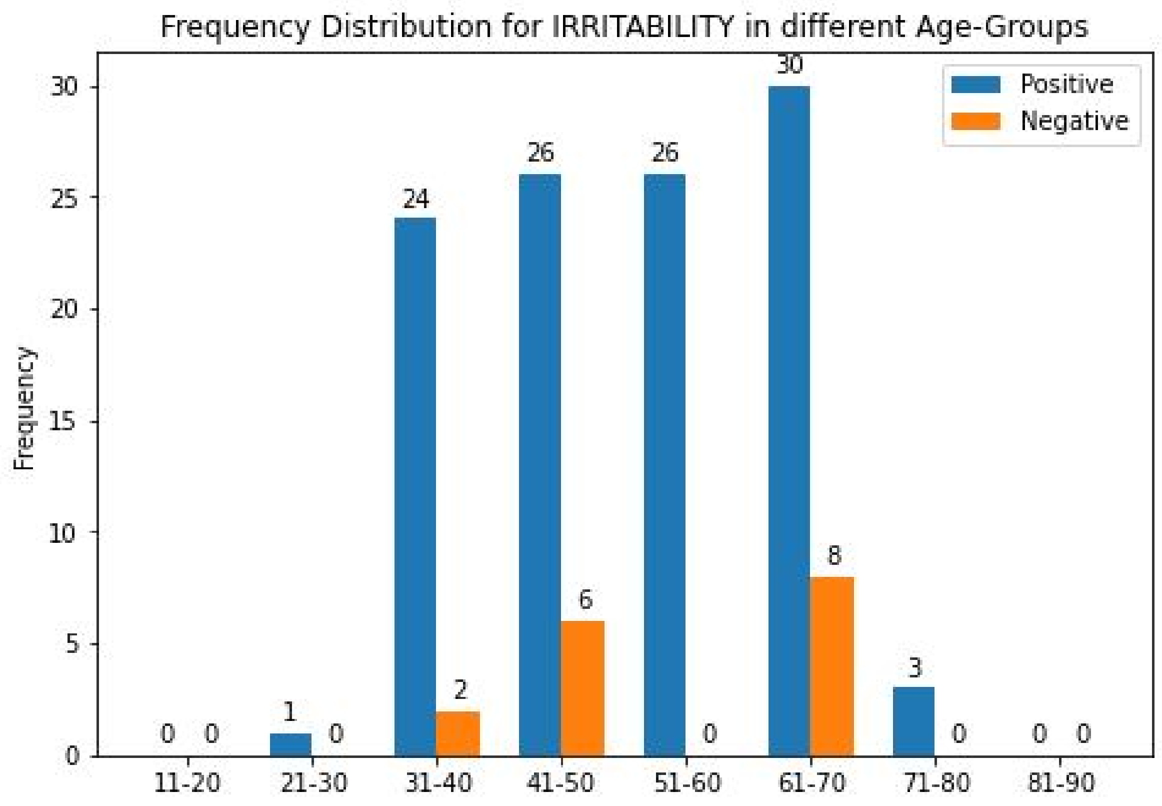
It can be inferred that people with **VISUAL BLURRING** are having high chances to have diabetes. This Symptom is having high occurrence in people among the age-group **31-70**. Also there are some cases with negative diabetes reports in spite of showing the symptom, especially in age-group **51-70**.

3.8 Itching



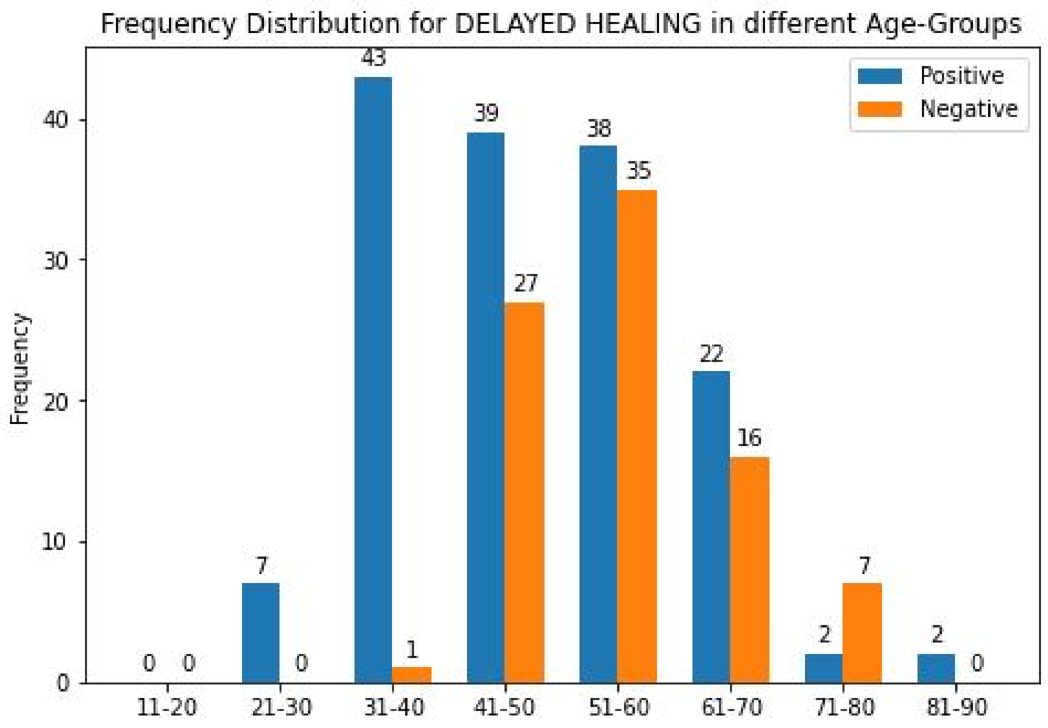
It can be inferred that people with **ITCHING** are having high chances to have diabetes. This Symptom is having high occurrence in people among the age-group **31-60**. Also there are some cases with negative diabetes reports in spite of showing the symptom.

3.9 Irritability



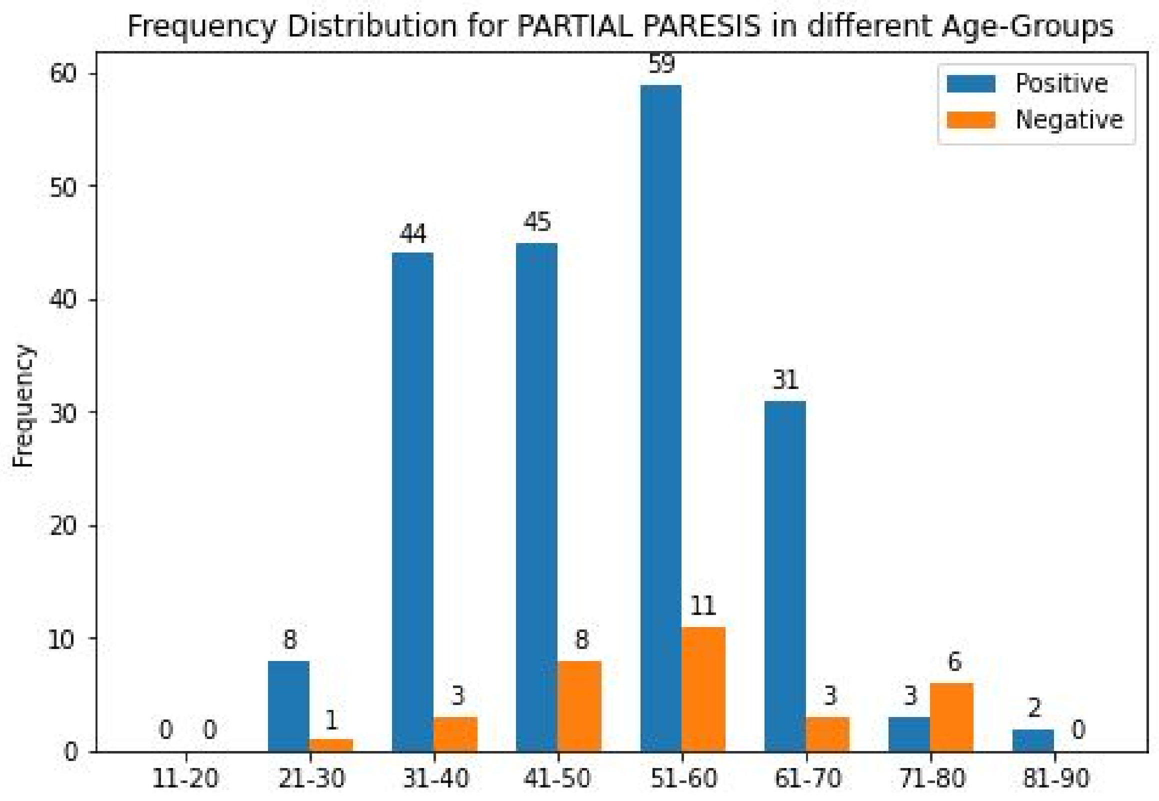
It can be inferred that people with **IRRITABILITY** are having high chances to have diabetes. This Symptom is having high occurrence in people among the age-group **31-70**. There are a few cases with negative diabetes reports in spite of showing the symptom.

3.10 Delayed Healing



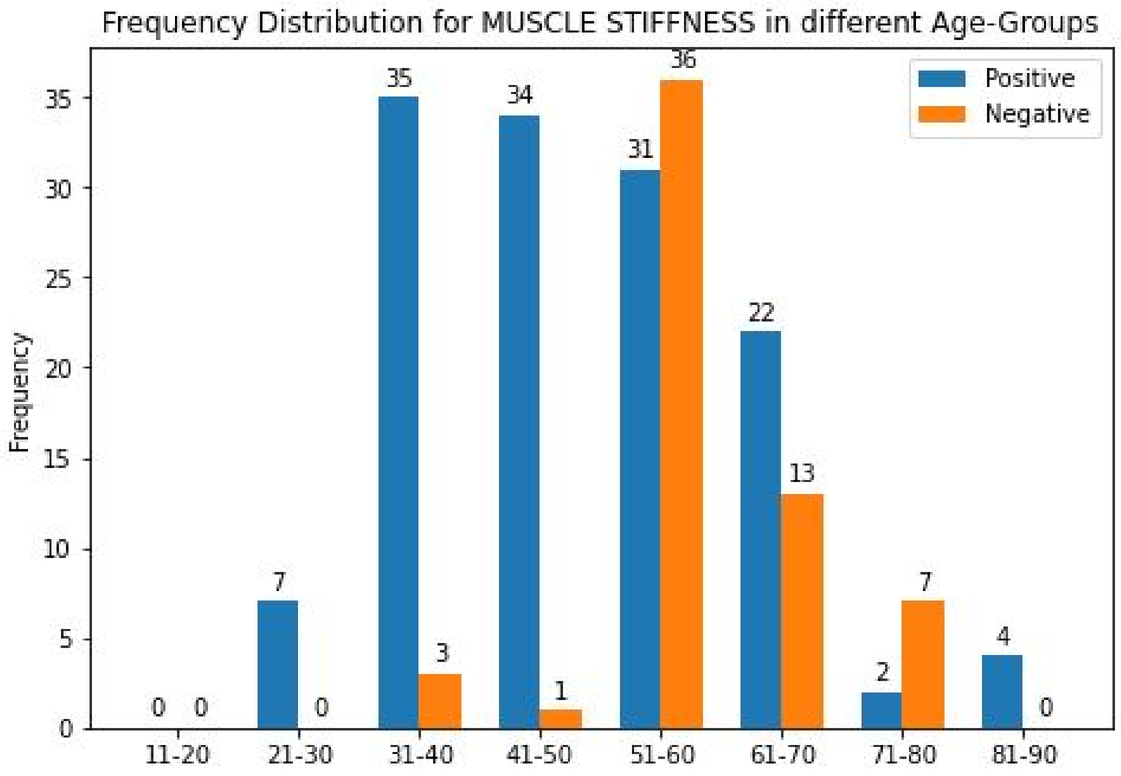
It can be inferred that people with **DELAYED HEALING** are having high chances to have diabetes. This Symptom is having high occurrence in people among the age-group **31-60**. There are very high number cases with negative diabetes reports in spite of showing the symptom, especially in age-group **41-70**.

3.11 Partial Paresis



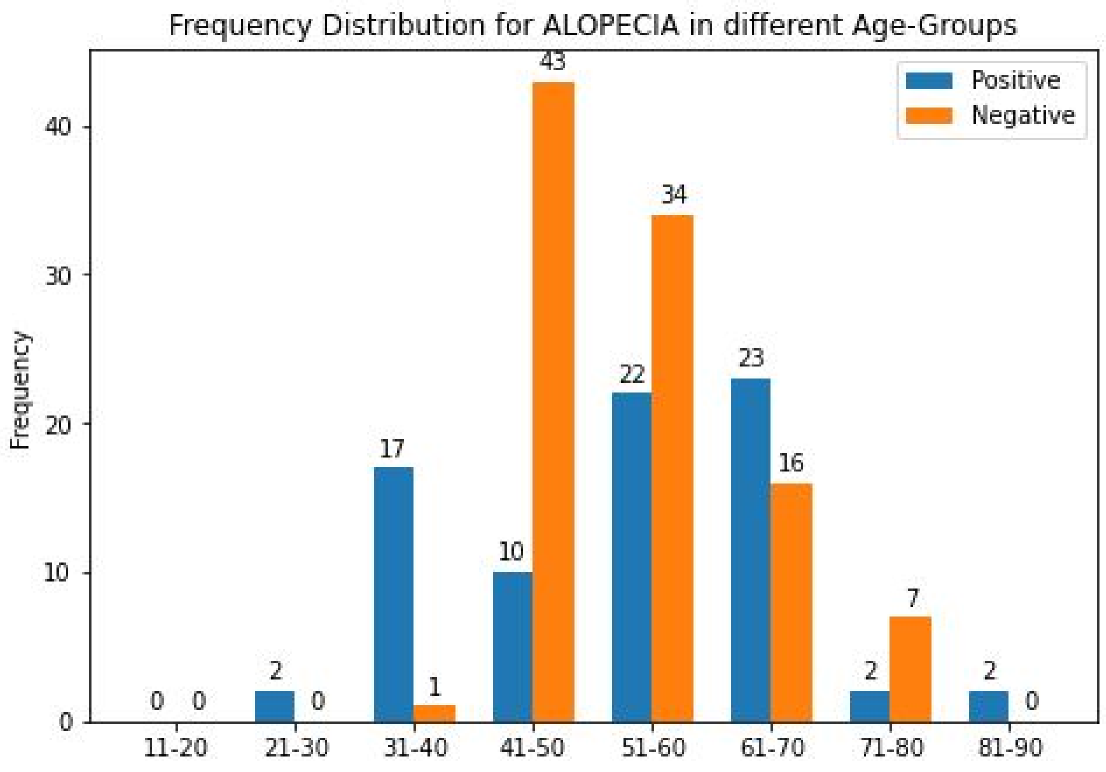
It can be inferred that people with **PARTIAL PARESIS** are having high chances to have diabetes. This Symptom is having high occurrence in people among the age-group **31-70**. Also there are some cases with negative diabetes reports in spite of showing the symptom. Cases with this symptom and negative report are high in age-group **71-80.**

3.12 Muscle Stiffness



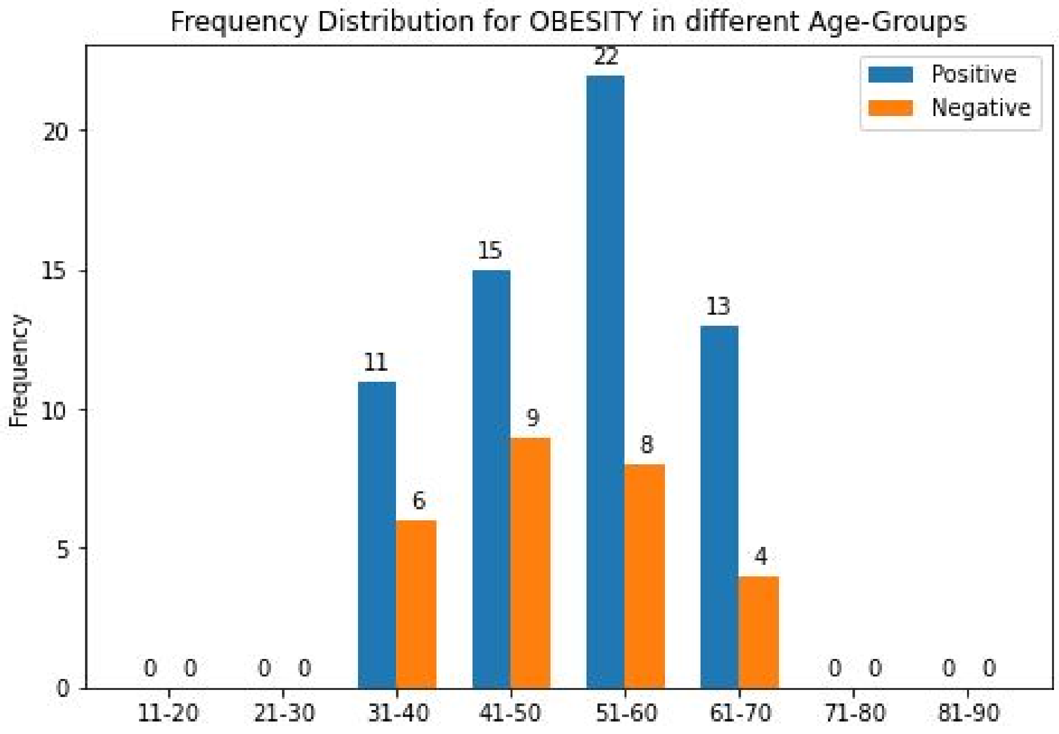
It can be inferred that people with **MUSCLE STIFFNESS** are having good chances to have diabetes. This Symptom is having high occurrence in people among the age-group **31-60**. There are a high number of cases where this symptom exists in non-diabetic patients. This is not a reliable symptom to include in the process of generating diabetes reports for age-group **51-60** where the number of cases with negative diabetes report are greater than number of cases with positive report.

3.13 Alopecia



It can be inferred that people with **ALOPECIA** are having good chances to have diabetes. This Symptom is having good occurrence in people among the age-group **51-70**. There are a very high number of cases where this symptom exists in non-diabetic patients, maily in age-group **41-80**. This is not a reliable symptom to include in the process of generating a diabetes report as the number of cases with negative diabetes report are far more than the number of cases with positive diabetes report in age-group **41-80**.

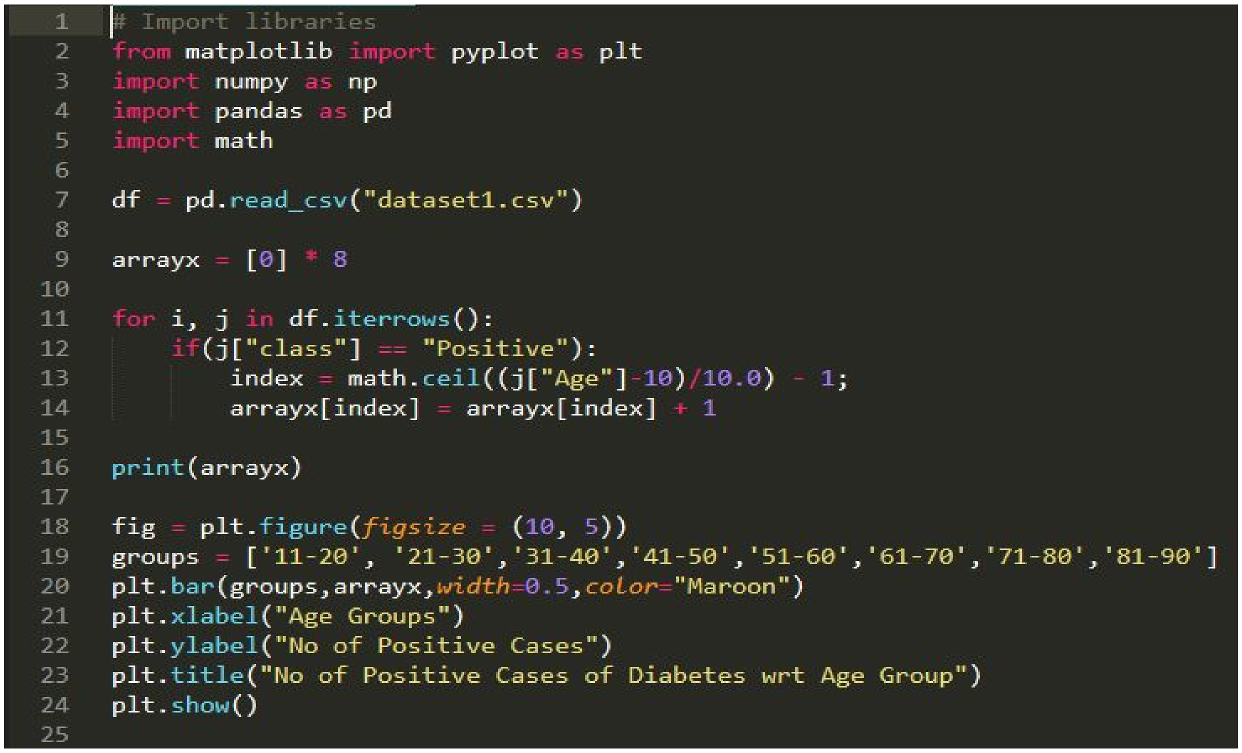
3.14 Obesity



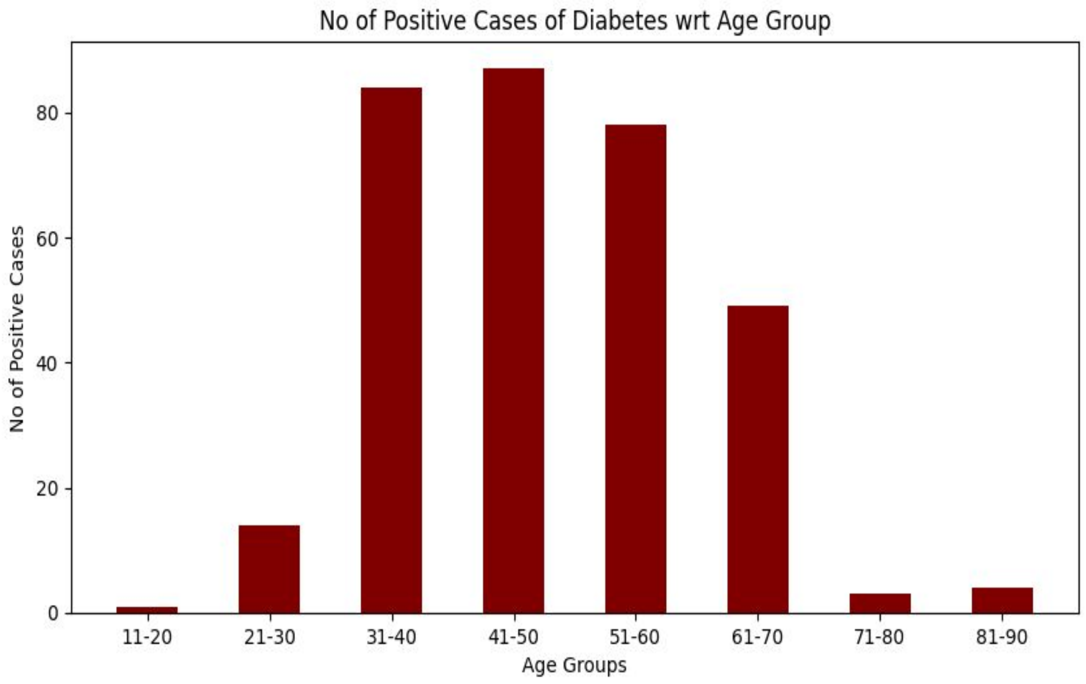
It can be inferred that people with **OBESITY** are having high chances to have diabetes. This Symptom is having high occurrence in people among the age-group **31-70**. Also there are some cases with negative diabetes reports in spite of showing the symptom.

Question Thought: Which age group has the highest number of cases of diabetes?

4. No of Positive Cases of Diabetes in different Age Groups



```python
# Import libraries
from matplotlib import pyplot as plt
import numpy as np
import pandas as pd
import math

df = pd.read_csv("dataset1.csv")

arrayx = [0] * 8

for i, j in df.iterrows():
    if(j["class"] == "Positive"):
        index = math.ceil((j["Age"]-10)/10.0) - 1;
        arrayx[index] = arrayx[index] + 1

print(arrayx)

fig = plt.figure(figsize = (10, 5))
groups = ['11-20', '21-30','31-40','41-50','51-60','61-70','71-80','81-90']
plt.bar(groups,arrayx,width=0.5,color="Maroon")
plt.xlabel("Age Groups")
plt.ylabel("No of Positive Cases")
plt.title("No of Positive Cases of Diabetes wrt Age Group")
plt.show()

```



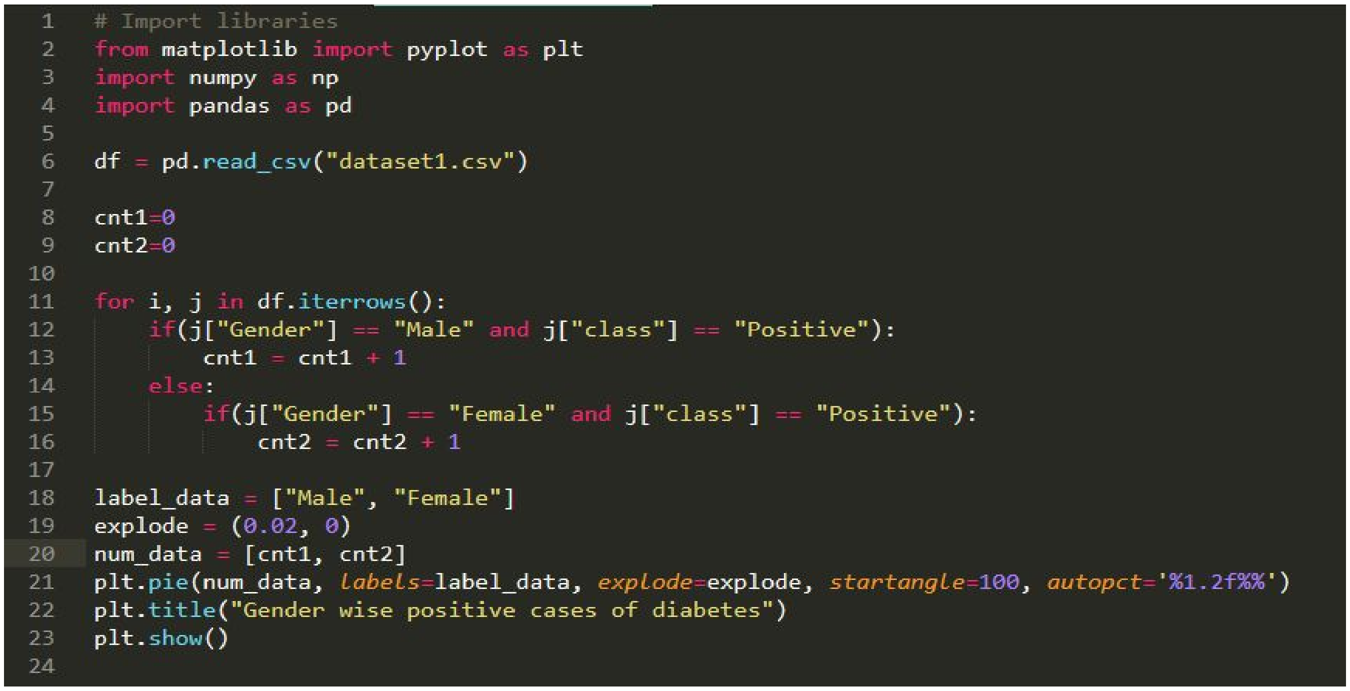
Age Groups: 11-20, 21-30, 31-40, 41-50, 51-60, 61-70, 71-80, 81-90

From the plotted bar graph we can conclude that age group 41-50 is maximally prone to Diabetes, followed by age groups 31-40 and 51-60. Overall, middle-aged people have more chances of being diabetic as compared to young or old people.

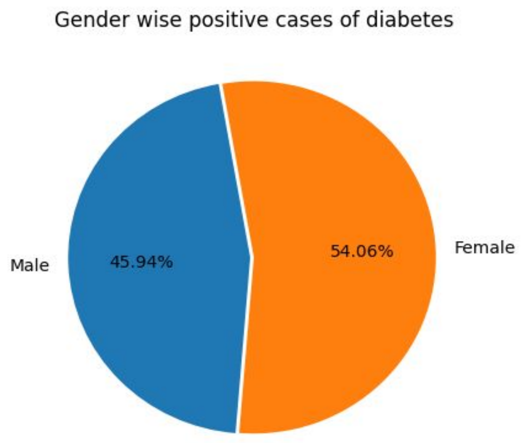
Also, young children and teenagers are significantly immune to diabetes.

Question Thought: Which gender has more no. diabetes cases and therefore is more prone to diabetes ?

5. Gender wise distribution of Diabetic people



```python
# Import libraries
from matplotlib import pyplot as plt
import numpy as np
import pandas as pd

df = pd.read_csv("dataset1.csv")

cnt1=0
cnt2=0

for i, j in df.iterrows():
    if(j["Gender"] == "Male" and j["class"] == "Positive"):
        cnt1 = cnt1 + 1
    else:
        if(j["Gender"] == "Female" and j["class"] == "Positive"):
            cnt2 = cnt2 + 1

label_data = ["Male", "Female"]
explode = (0.02, 0)
num_data = [cnt1, cnt2]
plt.pie(num_data, labels=label_data, explode=explode, startangle=100, autopct='%1.2f%%')
plt.title("Gender wise positive cases of diabetes")
plt.show()

```



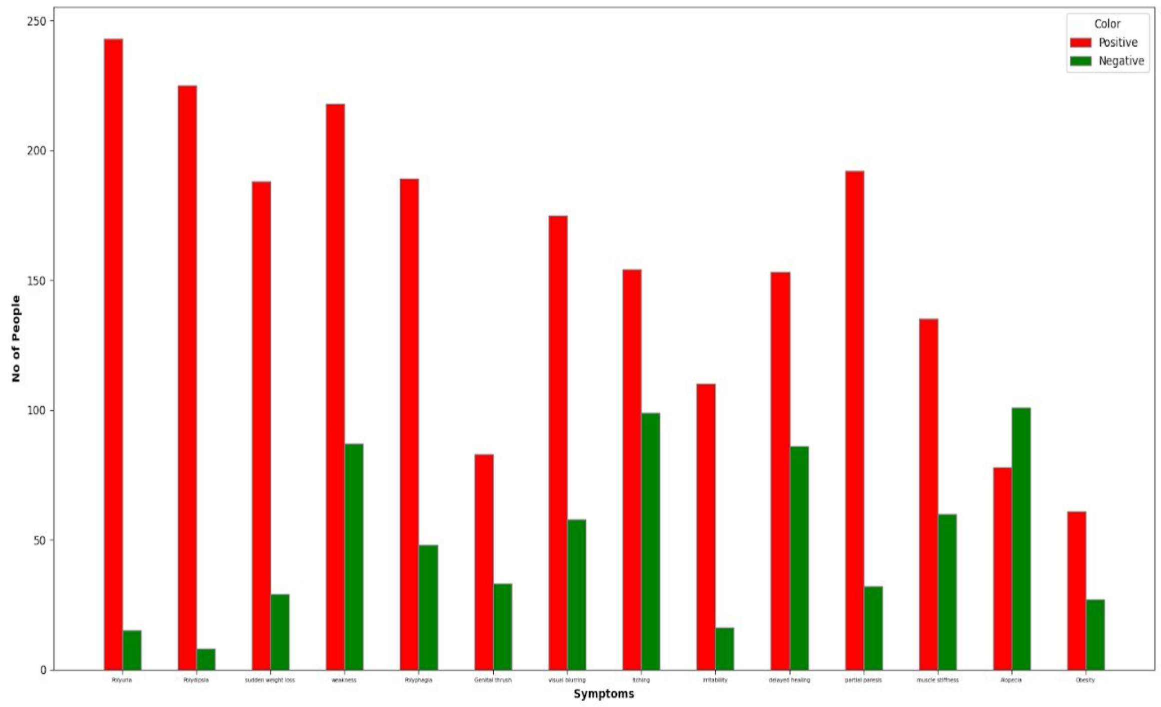
From the pie chart, we can infer that **females are more prone to diabetes than males**.

Question Thought: Which symptom is more important/mostly common to detect the presence of diabetes ?

6. Symptomatic distribution

```
import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

# set width of bar

barWidth = 1

fig = plt.subplots(figsize =(12, 8))



df = pd.read_csv("dataset1.csv")



Dict1 = {"Polyuria":0, "Polydipsia":0, "sudden weight loss":0, "weakness":0,
"Polyphagia":0, "Genital thrush":0, "visual blurring":0, "Itching":0,
"Irritability":0, "delayed healing":0, "partial paresis":0, "muscle
stiffness":0, "Alopecia":0, "Obesity":0}

Dict2 = {"Polyuria":0, "Polydipsia":0, "sudden weight loss":0, "weakness":0,
"Polyphagia":0, "Genital thrush":0, "visual blurring":0, "Itching":0,
"Irritability":0, "delayed healing":0, "partial paresis":0, "muscle
stiffness":0, "Alopecia":0, "Obesity":0}



for i, j in df.iterrows():

    for key in Dict1:

        if(j[key] == "Yes" and j["class"] == "Positive"):

            Dict1[key] = Dict1[key] + 1
        else:

            if(j[key] == "Yes" and j["class"] == "Negative"):

                Dict2[key] = Dict2[key] + 1




# Set position of bar on X axis

x1 = []

x2 = []

for key in Dict1:

    x1.append(Dict1[key])


for key in Dict2:

    x2.append(Dict2[key])


br1 = [1,5,9,13,17,21,25,29,33,37,41,45,49,53]

br2 = [x + barWidth for x in br1]


# Make the plot

p1 = plt.bar(br1, x1, color ='r', width = barWidth,

        edgecolor ='grey', label ='Positive')

p2 = plt.bar(br2, x2, color ='g', width = barWidth,

        edgecolor ='grey', label ='Negative')


x3 = [1.5,5.5,9.5,13.5,17.5,21.5,25.5,29.5,33.5,37.5,41.5,45.5,49.5,53.5]
```

```
# Adding Xticks

plt.xlabel('Symptoms', fontweight ='bold')

plt.ylabel('No of People', fontweight ='bold')

plt.xticks(x3,

       ["Polyuria","Polydipsia","sudden weight
loss","weakness","Polyphagia","Genital thrush","visual
blurring","Itching","Irritability","delayed healing","partial paresis","muscle
stiffness","Alopecia","Obesity"],

       fontsize=5)

plt.legend(handles=[p1, p2], title='Color')



plt.show()
```



This graph depicts that with each symptom present, how many people are actually diabetic, and how many aren’t.

Order of symptoms: "Polyuria","Polydipsia","sudden weight loss","weakness","Polyphagia","Genital thrush","visual blurring","Itching","Irritability","delayed healing","partial paresis","muscle stiffness","Alopecia","Obesity"

For each symptom the color denote:

RED: People that are diabetic

GREEN: People that are not diabetic

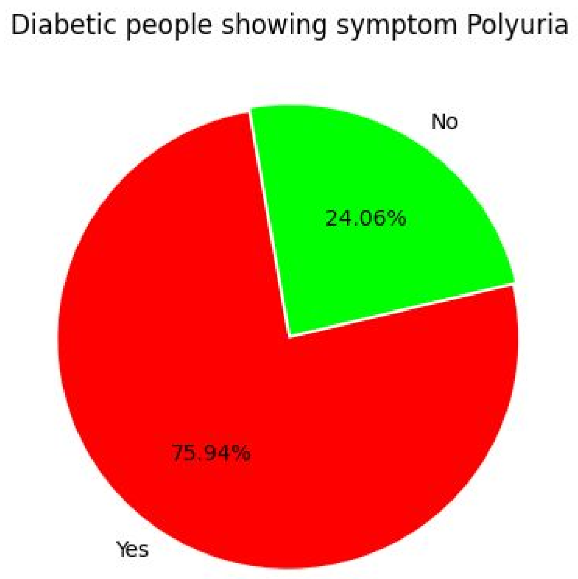
We can clearly see that the majority of people having ‘Alopecia’ are not diabetic, whereas symptoms ‘Polyuria’ and ‘Polydipsia’ seem strong enough that approximately all people having this symptom are diabetic.

Question Thought: Which Symptoms are not very significant or frequent in diabetes positive patients?

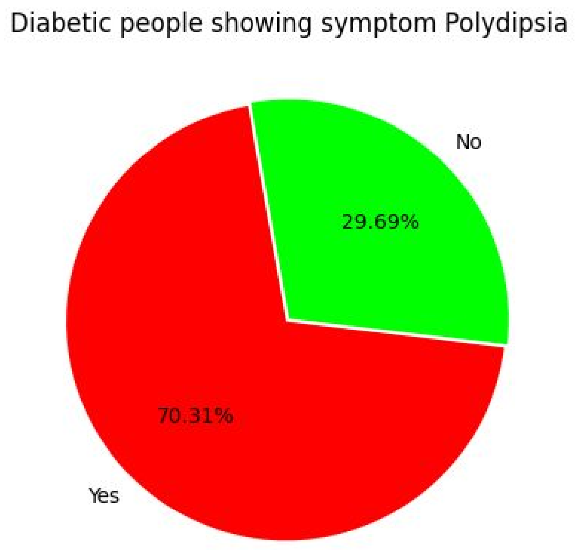
7. Symptom wise distribution of diabetic people

```
# Import libraries

from matplotlib import pyplot as plt

import numpy as np

import pandas as pd



df = pd.read_csv("dataset1.csv")

symptoms = ["Polyuria","Polydipsia","sudden weight
loss","weakness","Polyphagia","Genital thrush","visual
blurring","Itching","Irritability","delayed healing","partial paresis","muscle
stiffness","Alopecia","Obesity"]



def fun(col):

    cnt1=0

    cnt2=0



    for i, j in df.iterrows():

        if(j[col] == "Yes" and j["class"] == "Positive"):

            cnt1 = cnt1 + 1

        else:

            if(j[col] == "No" and j["class"] == "Positive"):

                cnt2 = cnt2 + 1
    label_data = ["Yes", "No"]

    explode = (0.02, 0)

    colors = ["red","lime"]

    num_data = [cnt1, cnt2]

    plt.pie(num_data, labels=label_data, colors=colors, explode=explode,
startangle=100, autopct='%1.2f%%')

    plt.title("Diabetic people showing symptom " + col)

    plt.show()


for s in symptoms:

    fun(s)
```

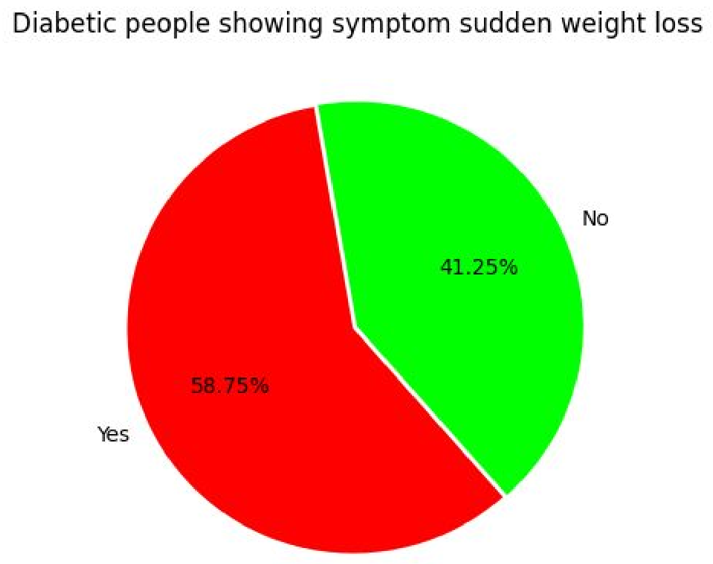
7.1 Polyuria



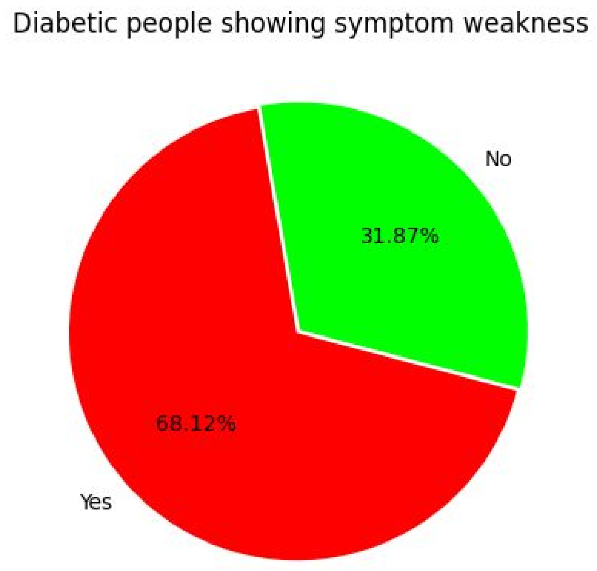
7.2 Polydipsia



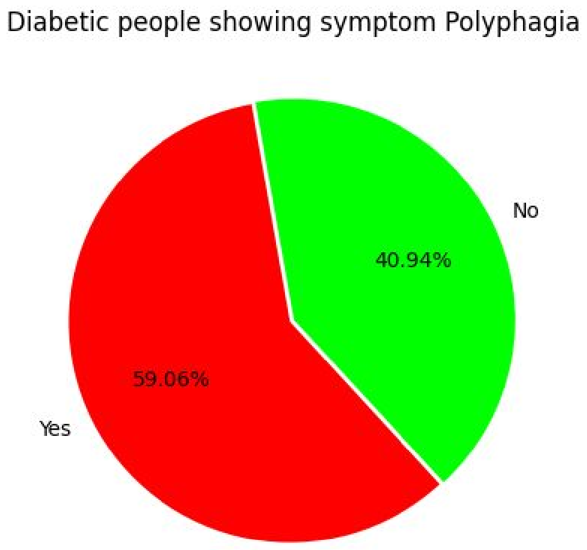
7.3 Sudden Weight Loss



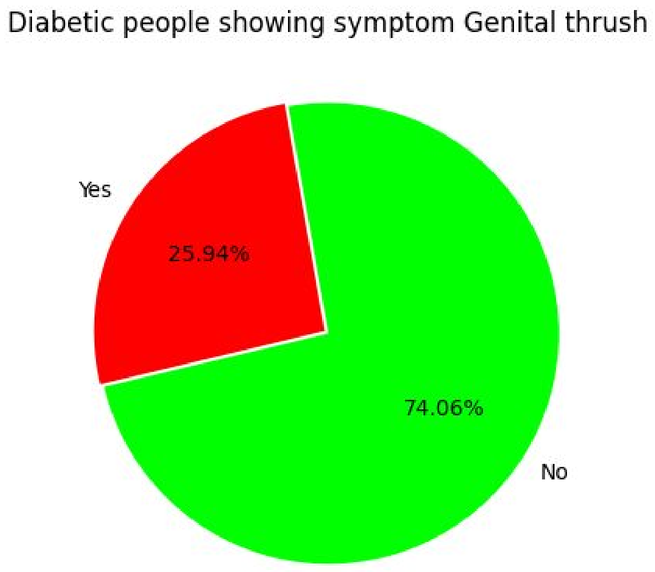
7.4 Weakness



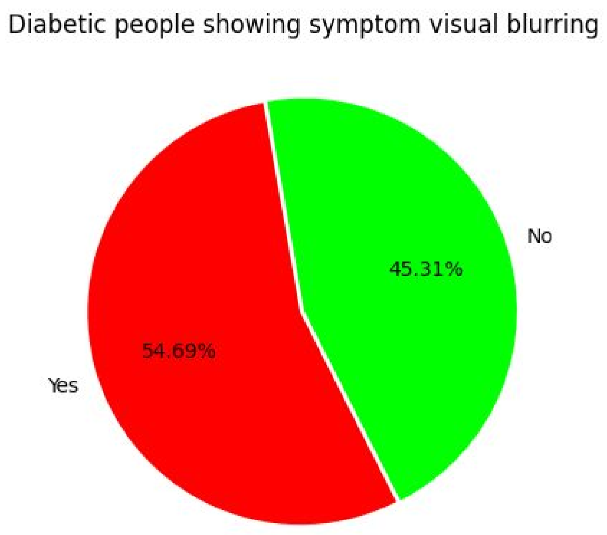
7.5 Polyphagia



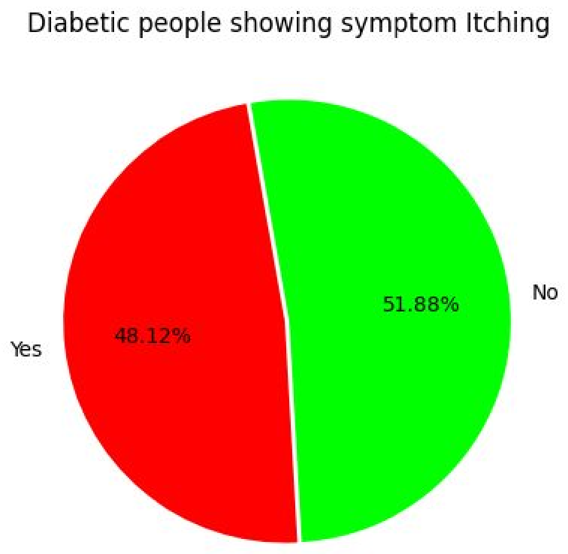
7.6 Genital thrush



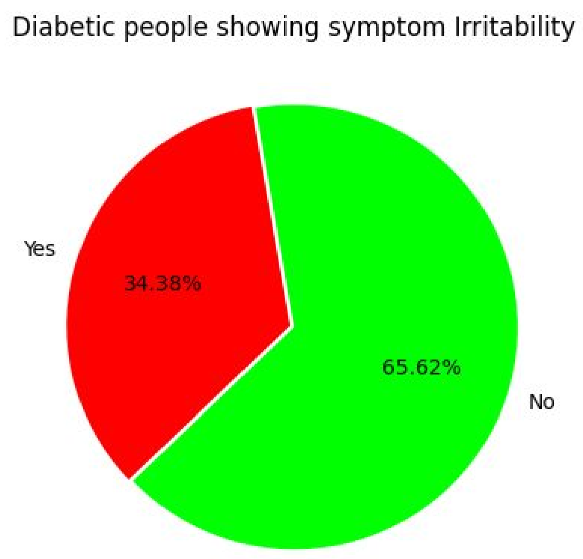
7.7 Visual blurring



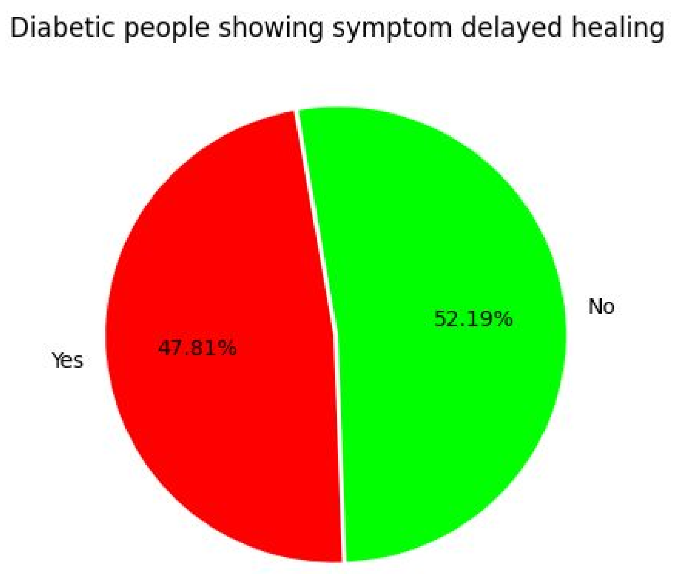
7.8 Itching



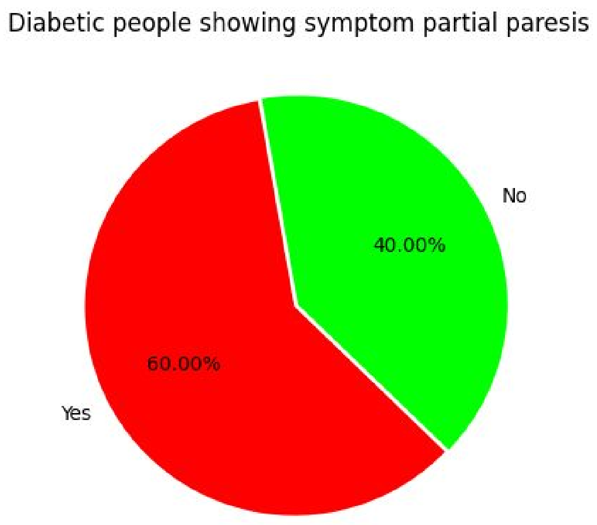
7.9 Irritability



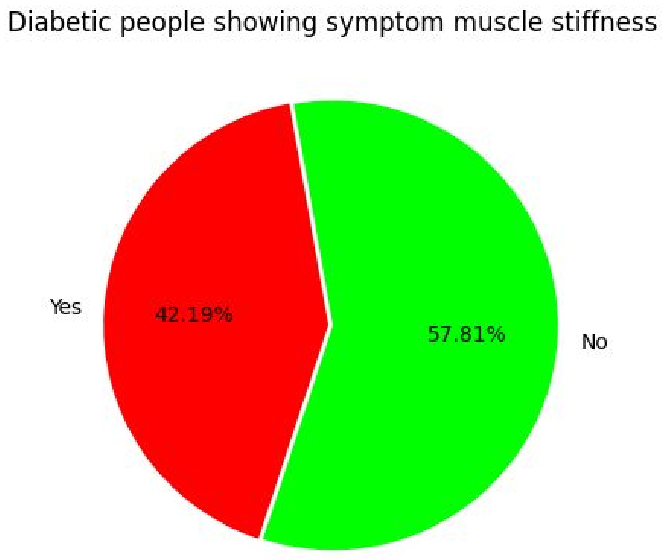
7.10 Delayed Healing



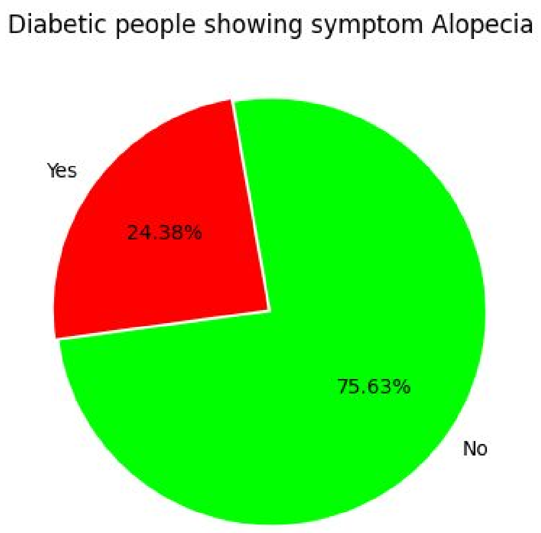
7.11 Partial Paresis



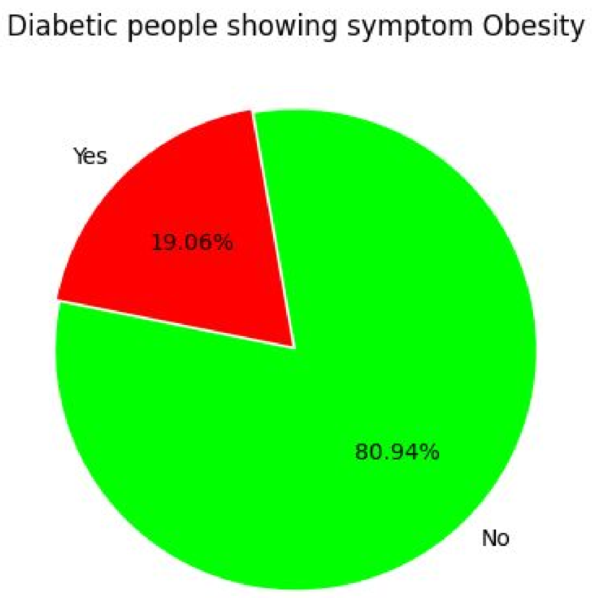
7.12 Muscle Stiffness



7.13 Alopecia



7.14 Obesity



From the above pie charts, we can infer that majority of diabetic people do not suffer from the following symptoms:

1. Obesity
2. Alopecia
3. Muscle stiffness
4. Delayed Healing
5. Irritability
6. Itching
7. Genital thrush

This does not mean that people having these symptoms are less likely to be diabetic but it means that the people who are diabetic do not show these symptoms majorly.

Implementation

The algorithm with an explanation of the algorithm used and comparison between both the algorithms, accuracy of the classifier, packages used, and their explanation.

ML Classification Algorithms

We are using **Gaussian Naive bayes algorithm and Decision tree algorithm** to classify the person as diabetes positive or negative according to symptoms and classification already done in training dataset.

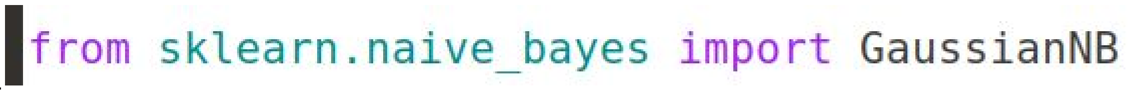
Why did we choose these two classifiers ?

Gaussian Naive Bayes Algorithm: Gaussian Naive Bayes classifier works on the famous bayes theorem. One of the assumptions which we require while using naive bayes algorithm is that attributes are conditionally independent which is true and can be assumed for our dataset as any one feature/attribute of data does not affect the value of another feature. In supervised learning Gaussian naive Bayes works quite efficiently and does not require a large dataset to train and learn the probability values required for prediction. We can use gaussian naive bayes as it has a simple design and implementation and unlike naive baye it does not take probability values to be zero for probability values which are not present in training dataset by assuming a normal distribution.The formula used to calculate the probability using the training dataset to find how likely a person is going to belong to a particular class label while classifying the test dataset is given below.

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

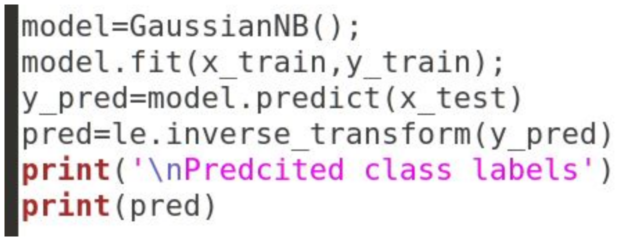
Decision Tree Algorithm : Decision Tree is one of the predictive modelling approaches. It uses a decision tree(as a predictive model) to conclude regarding the target values of an item(class of the item- leaf nodes), from observations about that item(represented by branches of the tree). Generally, a branch in a decision tree represents conjunction of various attributes and behaviour that lead to its classification to a particular class label as represented by the leaf nodes of the decision tree. And since our dataset required binary classification(classifying a person as diabetes positive or negative based on the symptoms), so we used Decision tree algorithm because it has a very simple design and to ensure best split of the attributes so as not to unnecessarily increase the depth of the tree we have made use of GINI index to determine the best attribute to split which further strengthens our algorithm to produce better classification results, in lesser time.

Implementation of Gaussian Naive Bayes algorithm



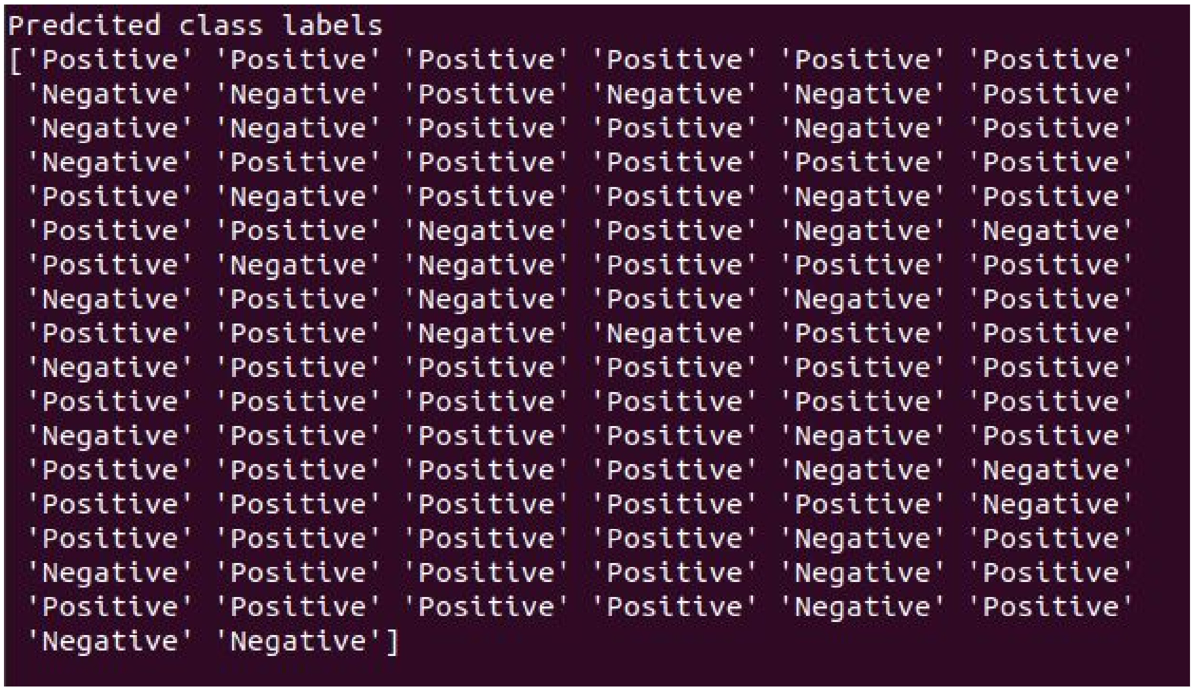
```
from sklearn.naive_bayes import GaussianNB
```

We used the sklearn library to import the Gaussian Naive Bayes classifier to predict the class of diabetes positive or negative by training the classifier on the already split dataset into training and testing parts.



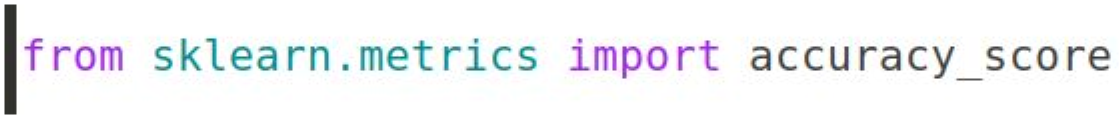
```
model=GaussianNB();
model.fit(x_train,y_train);
y_pred=model.predict(x_test)
pred=le.inverse_transform(y_pred)
print('\nPredcited class labels')
print(pred)
```

After importing the Gaussian Naive bayes classifier and preprocessing the dataset i.e. giving integer values to string values of features and splitting the dataset we are training the dataset in which X label is for the features like age and gender and symptoms of diabetes in this case while Y label is the class we need to predict that is whether the classifier predicts the person to be diabetes positive or negative.We again convert the integer values to string values by inverse_transform by importing preprocessing component from sklearn library

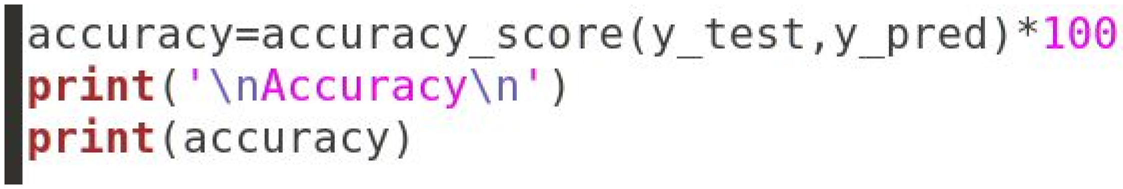


```
Predcited class labels
['Positive' 'Positive' 'Positive' 'Positive' 'Positive' 'Positive'
 'Negative' 'Negative' 'Positive' 'Negative' 'Negative' 'Positive'
 'Negative' 'Negative' 'Positive' 'Positive' 'Negative' 'Positive'
 'Negative' 'Positive' 'Positive' 'Positive' 'Positive' 'Positive'
 'Positive' 'Negative' 'Positive' 'Positive' 'Negative' 'Positive'
 'Positive' 'Positive' 'Negative' 'Positive' 'Negative' 'Negative'
 'Positive' 'Negative' 'Negative' 'Positive' 'Positive' 'Positive'
 'Negative' 'Positive' 'Negative' 'Positive' 'Negative' 'Positive'
 'Positive' 'Positive' 'Negative' 'Negative' 'Positive' 'Positive'
 'Negative' 'Positive' 'Positive' 'Positive' 'Positive' 'Positive'
 'Positive' 'Positive' 'Positive' 'Positive' 'Positive' 'Positive'
 'Negative' 'Positive' 'Positive' 'Positive' 'Negative' 'Positive'
 'Positive' 'Positive' 'Positive' 'Positive' 'Negative' 'Negative'
 'Positive' 'Positive' 'Positive' 'Positive' 'Positive' 'Negative'
 'Positive' 'Positive' 'Positive' 'Positive' 'Negative' 'Positive'
 'Negative' 'Positive' 'Positive' 'Positive' 'Negative' 'Positive'
 'Positive' 'Positive' 'Positive' 'Positive' 'Negative' 'Positive'
 'Negative' 'Negative']
```

The predicted class labels are as shown in the predicted class array which are predicted for the y labels for test data which is 20% of the entire data set i.e. 104 records . The class label predicted by the gaussian naive bayes classifier whether the person is diabetes positive or negative.

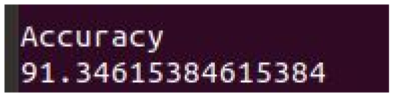


```
from sklearn.metrics import accuracy_score
```



```
accuracy=accuracy_score(y_test,y_pred)*100
print('\nAccuracy\n')
print(accuracy)
```

Accuracy for the Gaussian naive bayes algorithm is calculated by using the sklearn library from the accuracy_score component



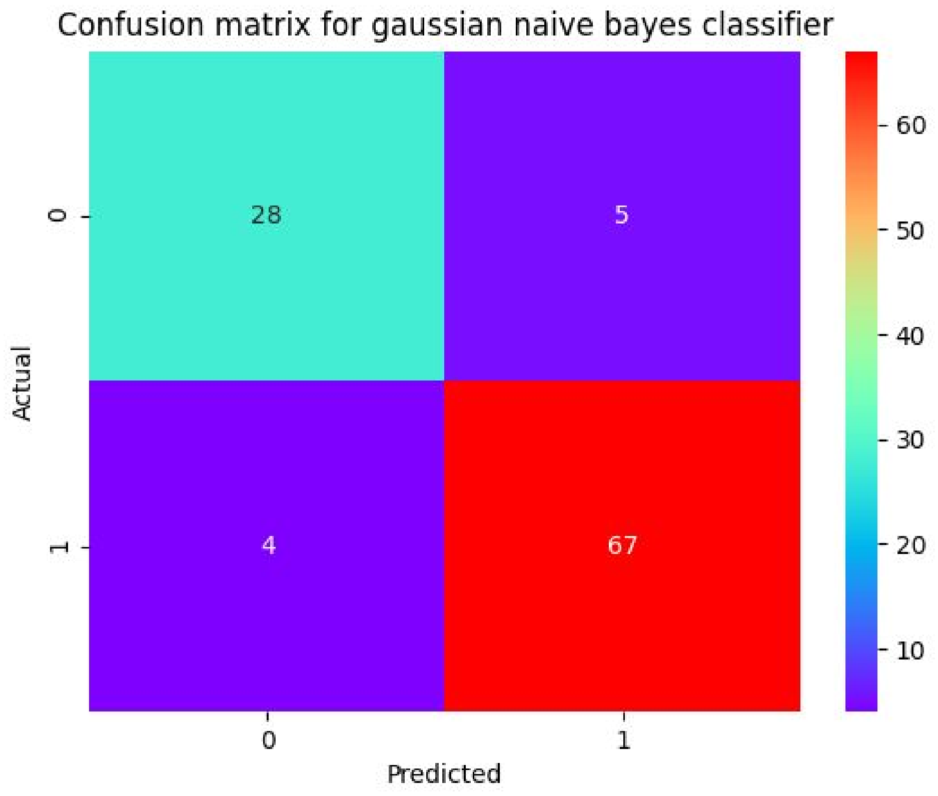
```
Accuracy
91.34615384615384
```

And the accuracy of Gaussian naive bayes classifier comes out to be 91.34% and out of 104 records in test data which is 20% of total data actually 71 records say positive and 33 are negative records and accuracy is better 68.26% which we would have got if the classifier would have been randomly guessing the class label **so we conclude that it is not a class imbalance problem** because even if we would have been guessing randomly always or always giving the same result irrespective of the values of the feature the accuracy would have been 68.26%.

```
from sklearn.metrics import confusion_matrix
import seaborn as sns  #for data visualization
import matplotlib.pyplot as plt  #for data visualization
```

```
m=confusion_matrix(y_test,y_pred)
sns.heatmap(confusion_matrix(y_test,y_pred),cmap='rainbow',annot=True,fmt="d")
#cmap value shows color for plot can be 'coolwarm', 'inferno', 'Blues','BuPu','Greens' etc.
plt.title('Confusion matrix for gaussian naive bayes classifier')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```

We use the sklearn.metrics library to import confusion_matrix to calculate the confusion matrix for are classifier and seaborn and matplotlib.pyplot to plot the confusion for visualization purpose.



Here 0 represents a Negative class label and 1 represents a Positive class label for classifying whether a person has diabetes or not. By analyzing the confusion matrix we see that it classified

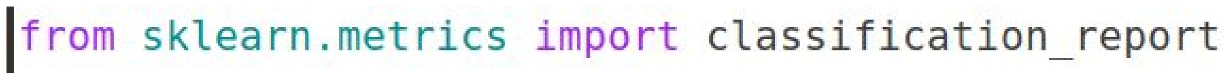
1.) True positive : 67 records
2.) True negative: 28 records
3.) False negative: 4 records
4.) False positive: 5 records

Accuracy = (TP+TN) / (TP+TN+FP+FN) = (67+28)/(67+28+4+5) = 0.9134

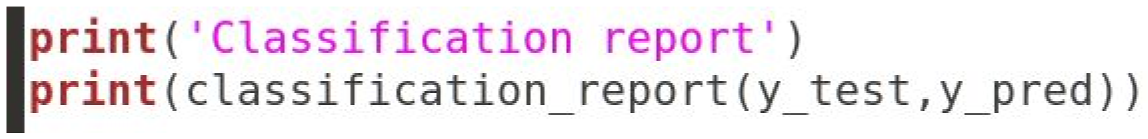
Therefore **Accuracy=91.34%** as we also calculated earlier by the use of sklearn library.

True positive rate (TPR) = TP/(TP+FN) = 67/ (67+4) = **0.94**

False positive rate (FPR) = FP/(FP+TN) = 5/(5+28) = **0.15**

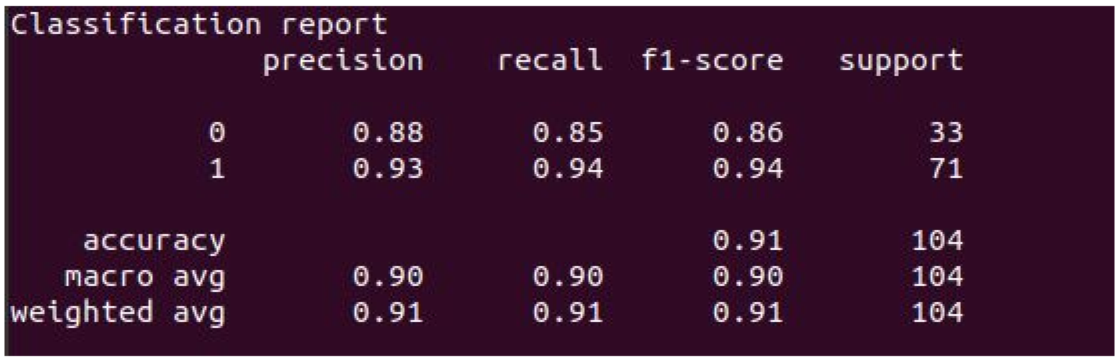


```
from sklearn.metrics import classification_report
```



```
print('Classification report')
print(classification_report(y_test,y_pred))
```

Again we use sklearn. metrics to import the classification report component so as to calculate the precision, recall, f score and support to find how good the classifier is and to compare the gaussian naive bayes classifier with the decision tree classifier we have used.



```
Classification report
              precision    recall  f1-score   support

           0       0.88      0.85      0.86        33
           1       0.93      0.94      0.94        71

    accuracy                           0.91       104
   macro avg       0.90      0.90      0.90       104
weighted avg       0.91      0.91      0.91       104
```

Therefore we see here the precision, f score, recall, support for positive class

Precision = Positive predicate value = (TP)/(TP+FP) = (67)/(67+5) = 0.93

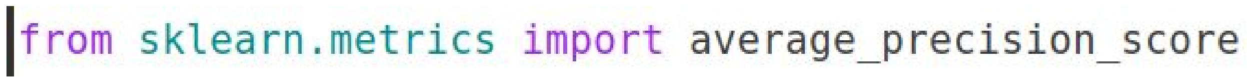
Recall = Sensitivity= TPR = 0.94

F1- score = (2*r*p)/(r+p) = 0.94

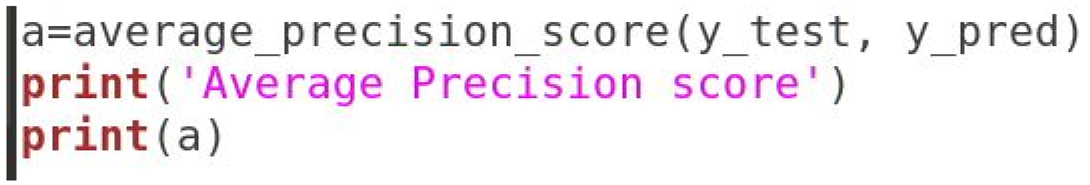
(r: Recall, p=Precision)

Support means that in the actual test data set there were 71 records with value as positive and 33 records as negative.

Also the value of precision, recall and f measure are near 1 which shows that the model is classifying test records quite efficiently.

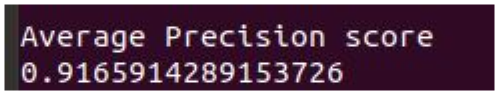


```
from sklearn.metrics import average_precision_score
```



```
a=average_precision_score(y_test, y_pred)
print('Average Precision score')
print(a)
```

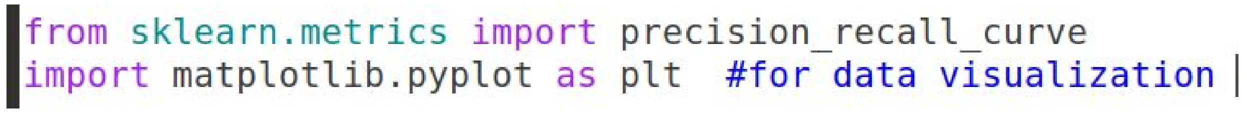
As the value of precision can change when the test set changes so using the average_precision_score component from sklearn.metrics library we calculate an average of the precision score.



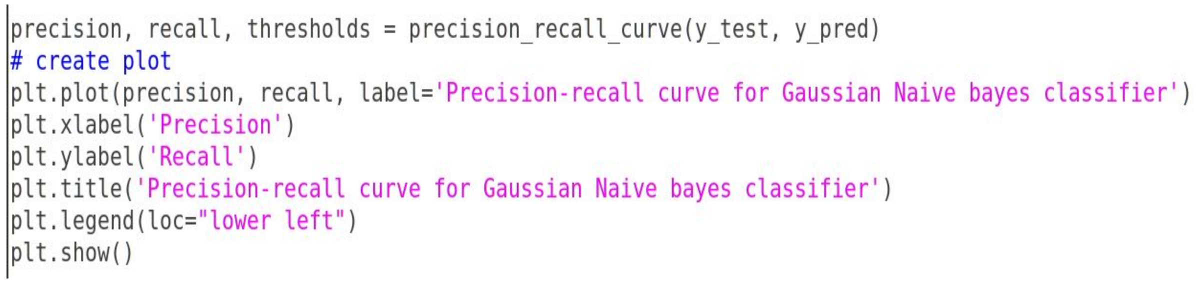
```
Average Precision score
0.9165914289153726
```

And the average precision of the Gaussian Naive bayes classifier comes out to be 0.91.

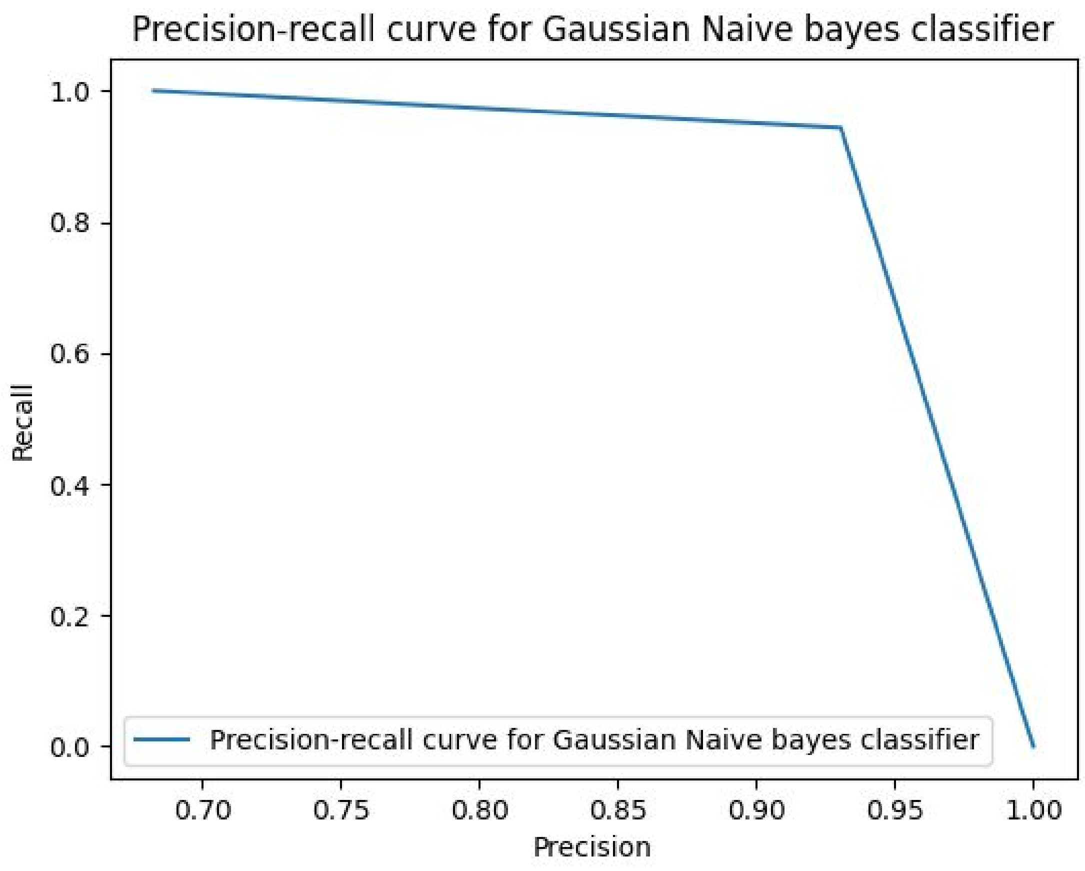
Also we can plot the precision vs recall curve for Gaussian naive bayes classifier sklearn library by importing precision_recall_curve component and using matplot library.



```
from sklearn.metrics import precision_recall_curve
import matplotlib.pyplot as plt  #for data visualization
```



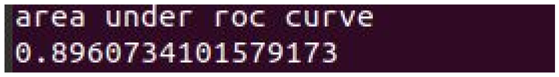
```
precision, recall, thresholds = precision_recall_curve(y_test, y_pred)
# create plot
plt.plot(precision, recall, label='Precision-recall curve for Gaussian Naive bayes classifier')
plt.xlabel('Precision')
plt.ylabel('Recall')
plt.title('Precision-recall curve for Gaussian Naive bayes classifier')
plt.legend(loc="lower left")
plt.show()
```



Precision-Recall curves summarize the trade-off between the true positive rate and the positive predictive value for a predictive model using different probability thresholds. A model with perfect skill is depicted as a point at (1,1). A skilful model is represented by a curve that bows towards (1,1) above the flat line of no skill. So we can see our model is also quite efficient. As it also plots a flat line.

```
from sklearn.metrics import roc_curve, auc
```

```
fpr, tpr,thresholds = roc_curve(y_test, y_pred)
ab=auc(fpr,tpr)
print ('area under roc curve')
print (ab)
```

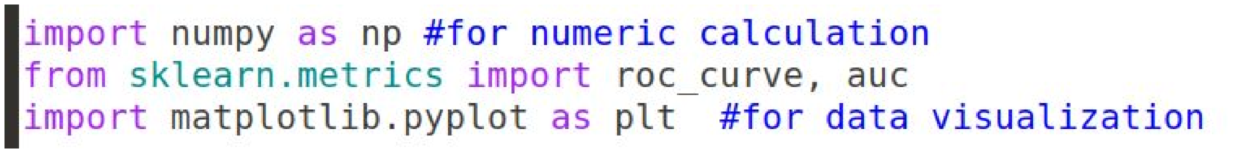


```
area under roc curve
0.8960734101579173
```

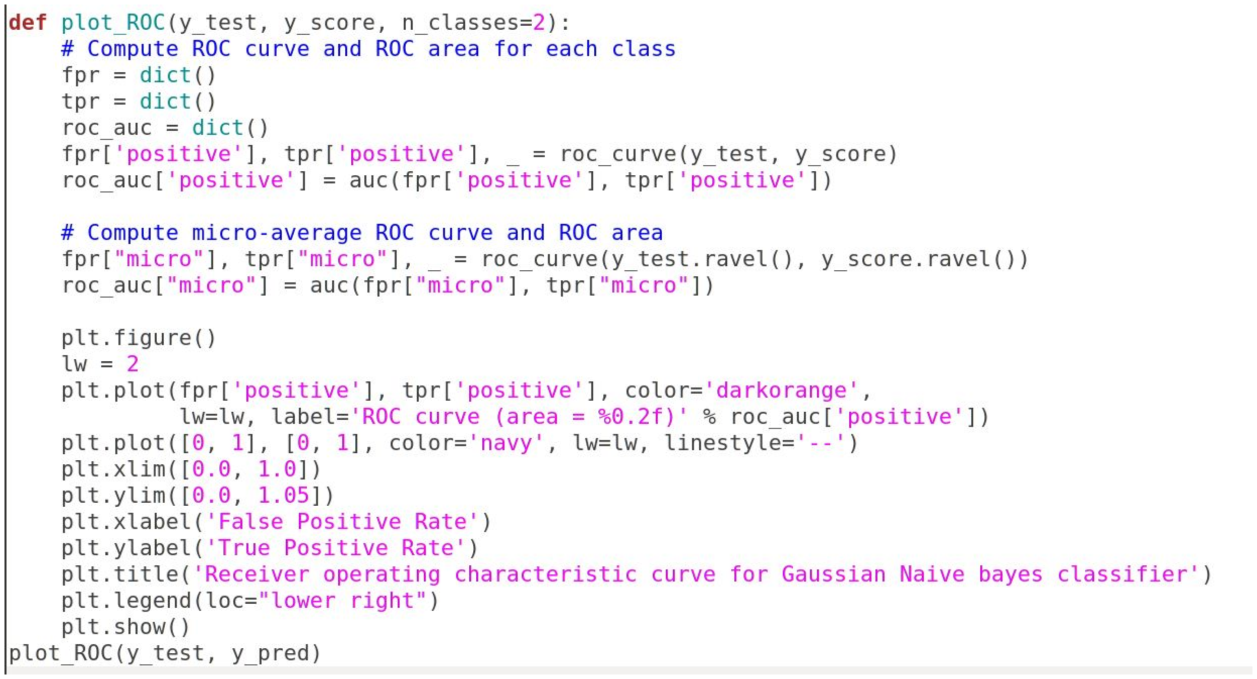
Area under ROC curve = 0.90

ROC (Receiver operating Characteristic) is more suitable for our data than precision recall curve to analyze our model as our data does not have a class imbalance problem.

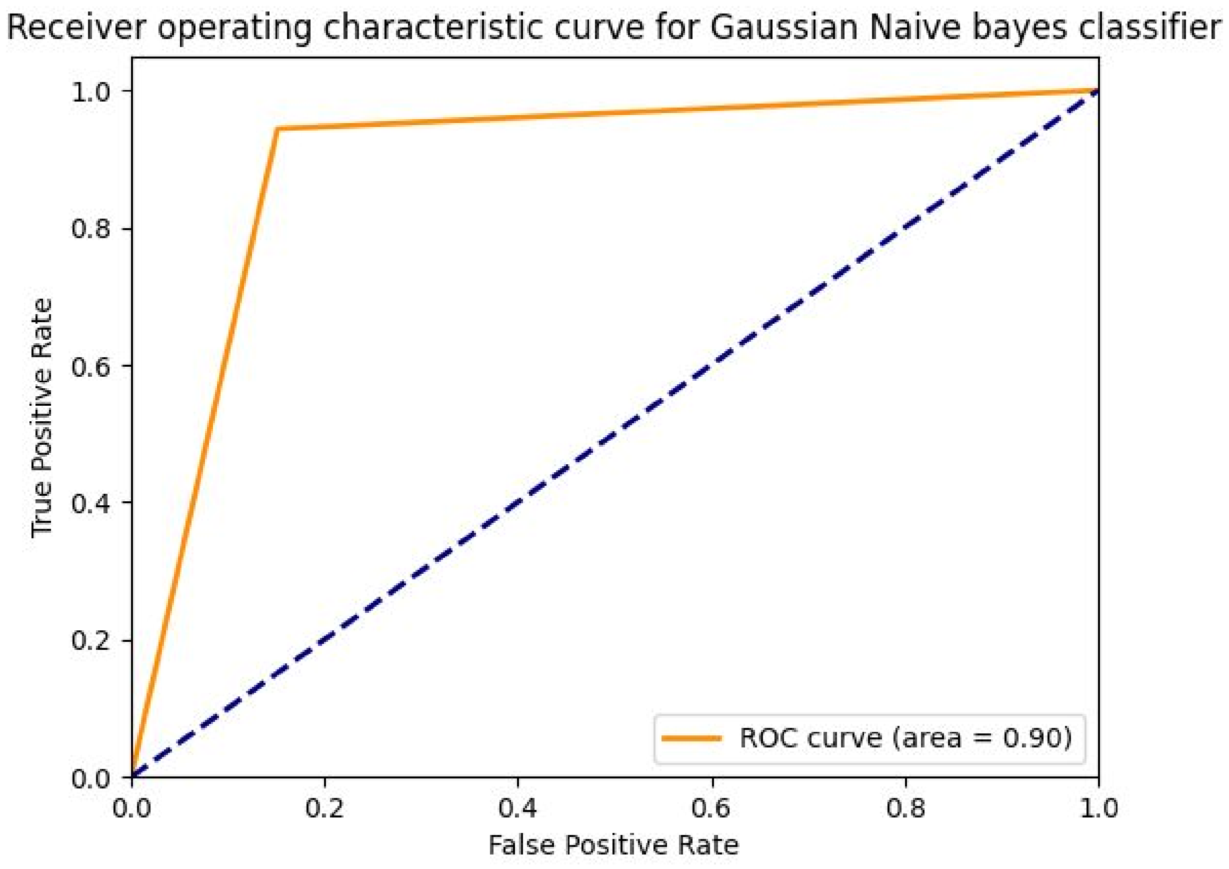
Area under the roc curve shows how good our classifier is as the ideal area under the curve should be 1. Using the roc_curve imported from sklearn.metrics library we have calculated that the area under the curve comes out to be 0.896 ~ 0.9 which is good as it is close to 1 so our model works quite efficiently.



```python
import numpy as np #for numeric calculation
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt  #for data visualization
```



```python
def plot_ROC(y_test, y_score, n_classes=2):
    # Compute ROC curve and ROC area for each class
    fpr = dict()
    tpr = dict()
    roc_auc = dict()
    fpr['positive'], tpr['positive'], _ = roc_curve(y_test, y_score)
    roc_auc['positive'] = auc(fpr['positive'], tpr['positive'])

    # Compute micro-average ROC curve and ROC area
    fpr["micro"], tpr["micro"], _ = roc_curve(y_test.ravel(), y_score.ravel())
    roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])

    plt.figure()
    lw = 2
    plt.plot(fpr['positive'], tpr['positive'], color='darkorange',
             lw=lw, label='ROC curve (area = %0.2f)' % roc_auc['positive'])
    plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver operating characteristic curve for Gaussian Naive bayes classifier')
    plt.legend(loc="lower right")
    plt.show()
plot_ROC(y_test, y_pred)
```



ROC curve plots True positive rate (TPR) on Y axis and False positive rate (FPR) on X axis and if the area under the curve is close to 1 it is an ideal condition as the ideal plot would be a straight line joining points (0,0) and (0,1). The diagonal line joins two extreme conditions of classifying all records as positive or all records as negative without classifying the records actually.

Gaussian Naive Bayes classifier turns out to be quite efficient for our data set as the area under the roc curve i.e. 0.9 is close to 1 and can be used as our data does not have class imbalance problems.

Implementation of Decision Tree algorithm

```
In [70]: from sklearn.tree import DecisionTreeClassifier
```

We used the sklearn library to import the Decision Tree classifier to predict the class of diabetes positive or negative by training the classifier on the already split dataset into training and testing parts.

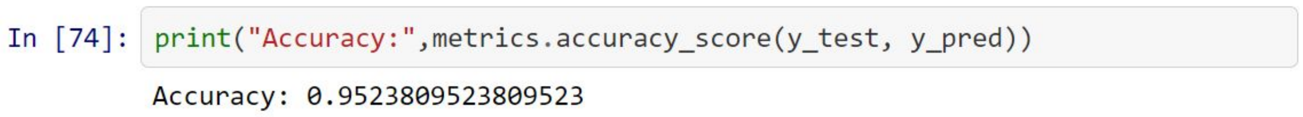
```
In [71]: clf = DecisionTreeClassifier()
         clf = clf.fit(X_train, y_train)
         y_pred = clf.predict(X_test)
```

After importing the Decision Tree classifier and preprocessing the dataset i.e. giving integer values to categorical attributes(having string values) and splitting the dataset we are training the dataset in which X label is for the attributes like age and gender and symptoms of diabetes in this case while Y label is the class we need to predict that is whether the classifier predicts the person to be diabetes positive or negative.

```
In [72]: print(y_pred)

         ['Negative' 'Positive' 'Negative' 'Negative' 'Positive' 'Positive'
          'Negative' 'Positive' 'Negative' 'Positive' 'Negative' 'Negative'
          'Positive' 'Positive' 'Negative' 'Positive' 'Negative' 'Negative'
          'Positive' 'Positive' 'Positive' 'Negative' 'Positive' 'Negative'
          'Positive' 'Negative' 'Positive' 'Positive' 'Negative' 'Positive'
          'Positive' 'Positive' 'Positive' 'Positive' 'Positive' 'Positive'
          'Positive' 'Negative' 'Negative' 'Positive' 'Negative' 'Negative'
          'Negative' 'Positive' 'Positive' 'Positive' 'Positive' 'Positive'
          'Positive' 'Positive' 'Negative' 'Positive' 'Positive' 'Positive'
          'Positive' 'Positive' 'Negative' 'Positive' 'Negative' 'Negative'
          'Positive' 'Negative' 'Positive' 'Positive' 'Positive' 'Negative'
          'Positive' 'Negative' 'Negative' 'Positive' 'Positive' 'Positive'
          'Positive' 'Positive' 'Negative' 'Positive' 'Positive' 'Positive'
          'Negative' 'Positive' 'Positive' 'Positive' 'Positive' 'Negative'
          'Positive' 'Negative' 'Negative' 'Positive' 'Positive' 'Positive'
          'Positive' 'Negative' 'Negative' 'Positive' 'Positive' 'Positive'
          'Positive' 'Positive' 'Positive' 'Positive' 'Positive' 'Positive'
          'Positive' 'Positive' 'Negative']
```

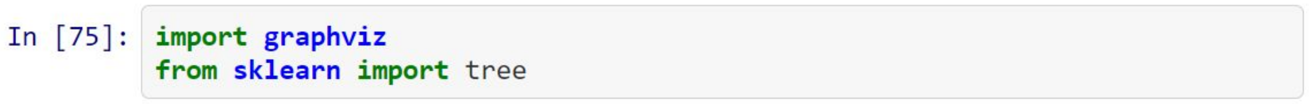
Printing the values predicted by the decision tree classifier.

```
In [73]: from sklearn import metrics
```



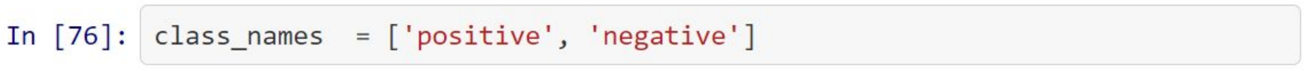
```
In [74]: print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

         Accuracy: 0.9523809523809523
```

Computing the accuracy of the decision tree algorithm by importing the sklearn.metrics library.

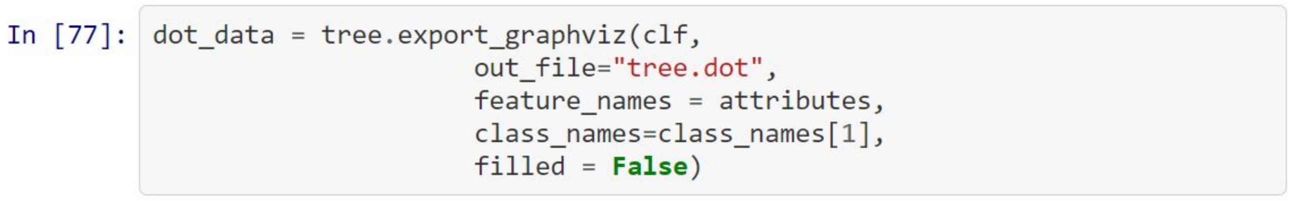


```
In [75]: import graphviz
         from sklearn import tree
```

Importing the graphviz library and tree library from the sklearn to visualize the decision tree formed, so as to give a proper view and exact calculation about the gini index calculate at each step and the criterion and attributes of splitting. Here we have considered binary splitting(i.e. Two way splitting of attributes).



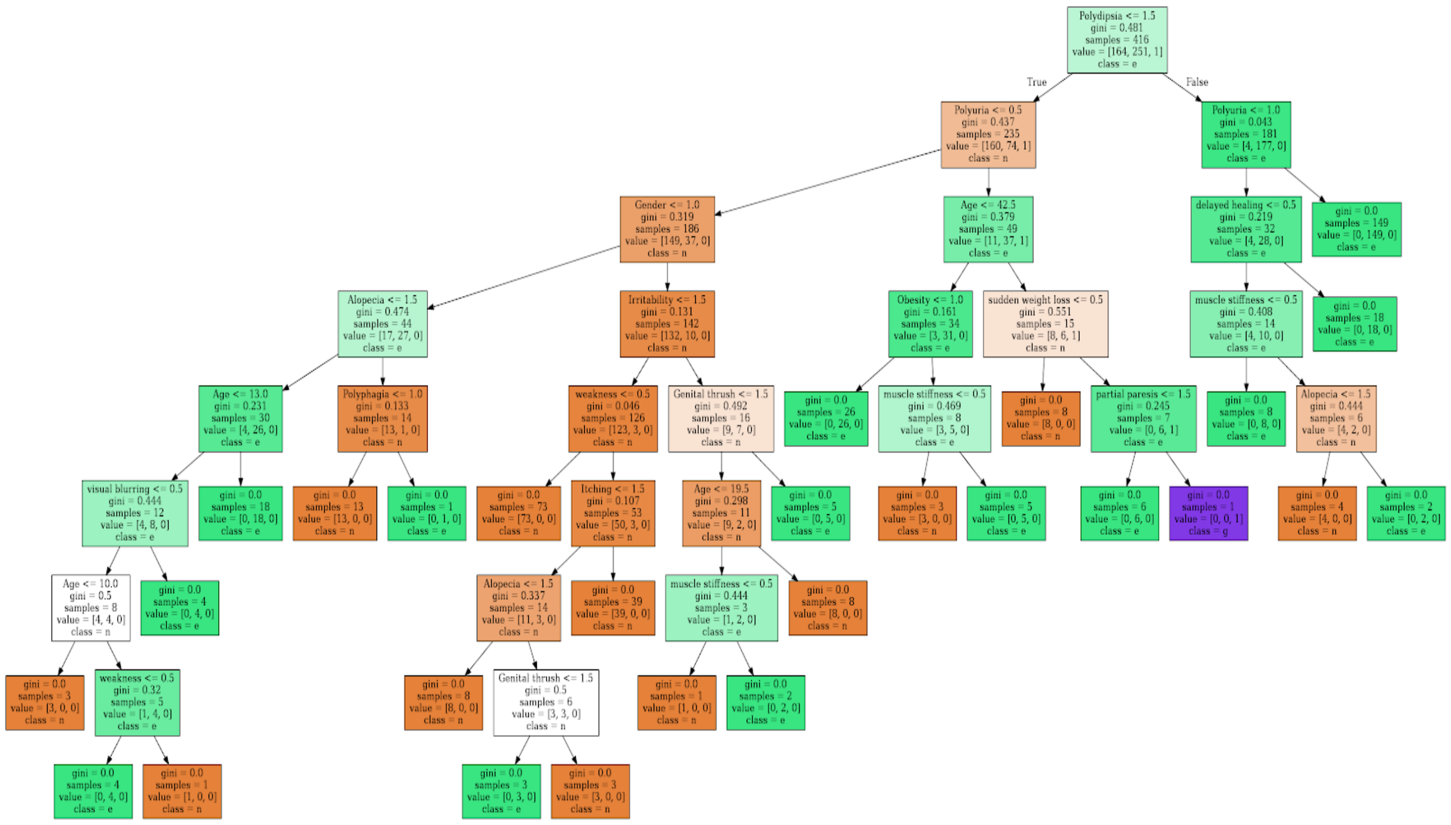
```
In [76]: class_names  = ['positive', 'negative']
```



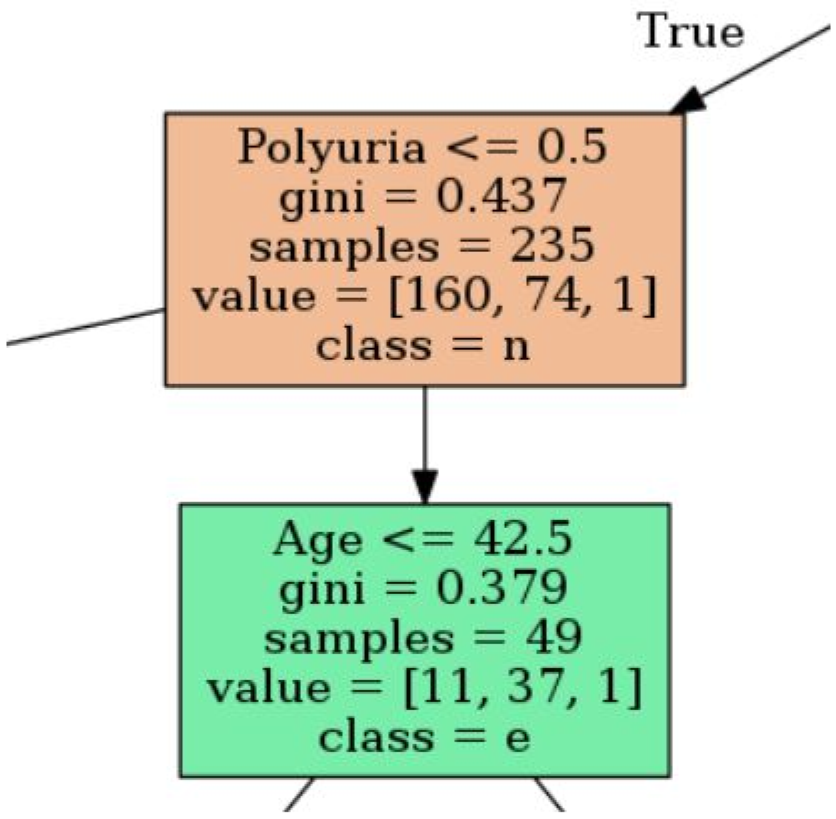
```
In [77]: dot_data = tree.export_graphviz(clf,
                                out_file="tree.dot",
                                feature_names = attributes,
                                class_names=class_names[1],
                                filled = False)
```

Exporting the tree so formed as a dot file and then later have converted it into png format to get a user-friendly view of the tree.

Decision tree depicting the split and Gini index at each node


Polydipsia <= 1.5 gini = 0.481 samples = 416 value = [164, 251, 1] class = e
True
False
Polyuria <= 0.5 gini = 0.437 samples = 235 value = [160, 74, 1] class = n
Polyuria <= 1.0 gini = 0.043 samples = 181 value = [4, 177, 0] class = e
Gender <= 1.0 gini = 0.319 samples = 186 value = [149, 37, 0] class = n
Age <= 42.5 gini = 0.379 samples = 49 value = [11, 37, 1] class = e
delayed healing <= 0.5 gini = 0.219 samples = 32 value = [4, 28, 0] class = e
gini = 0.0 samples = 149 value = [0, 149, 0] class = e
Alopecia <= 1.5 gini = 0.474 samples = 44 value = [17, 27, 0] class = e
Irritability <= 1.5 gini = 0.131 samples = 142 value = [132, 10, 0] class = n
Obesity <= 1.0 gini = 0.161 samples = 34 value = [3, 31, 0] class = e
sudden weight loss <= 0.5 gini = 0.551 samples = 15 value = [8, 6, 1] class = n
muscle stiffness <= 0.5 gini = 0.408 samples = 14 value = [4, 10, 0] class = e
gini = 0.0 samples = 18 value = [0, 18, 0] class = e
Age <= 13.0 gini = 0.231 samples = 30 value = [4, 26, 0] class = e
Polyphagia <= 1.0 gini = 0.133 samples = 14 value = [13, 1, 0] class = n
weakness <= 0.5 gini = 0.046 samples = 126 value = [123, 3, 0] class = n
Genital thrush <= 1.5 gini = 0.492 samples = 16 value = [9, 7, 0] class = n
gini = 0.0 samples = 26 value = [0, 26, 0] class = e
muscle stiffness <= 0.5 gini = 0.469 samples = 8 value = [3, 5, 0] class = e
gini = 0.0 samples = 8 value = [8, 0, 0] class = n
partial paresis <= 1.5 gini = 0.245 samples = 7 value = [0, 6, 1] class = e
gini = 0.0 samples = 8 value = [0, 8, 0] class = e
Alopecia <= 1.5 gini = 0.444 samples = 6 value = [4, 2, 0] class = n
visual blurring <= 0.5 gini = 0.444 samples = 12 value = [4, 8, 0] class = e
gini = 0.0 samples = 18 value = [0, 18, 0] class = e
gini = 0.0 samples = 13 value = [13, 0, 0] class = n
gini = 0.0 samples = 1 value = [0, 1, 0] class = e
gini = 0.0 samples = 73 value = [73, 0, 0] class = n
Itching <= 1.5 gini = 0.107 samples = 53 value = [50, 3, 0] class = n
Age <= 19.5 gini = 0.298 samples = 11 value = [9, 2, 0] class = n
gini = 0.0 samples = 5 value = [0, 5, 0] class = e
gini = 0.0 samples = 3 value = [3, 0, 0] class = n
gini = 0.0 samples = 5 value = [0, 5, 0] class = e
gini = 0.0 samples = 6 value = [0, 6, 0] class = e
gini = 0.0 samples = 1 value = [0, 0, 1] class = g
gini = 0.0 samples = 4 value = [4, 0, 0] class = n
gini = 0.0 samples = 2 value = [0, 2, 0] class = e
Age <= 10.0 gini = 0.5 samples = 8 value = [4, 4, 0] class = n
gini = 0.0 samples = 4 value = [0, 4, 0] class = e
Alopecia <= 1.5 gini = 0.337 samples = 14 value = [11, 3, 0] class = n
gini = 0.0 samples = 39 value = [39, 0, 0] class = n
muscle stiffness <= 0.5 gini = 0.444 samples = 3 value = [1, 2, 0] class = e
gini = 0.0 samples = 8 value = [8, 0, 0] class = n
gini = 0.0 samples = 3 value = [3, 0, 0] class = n
weakness <= 0.5 gini = 0.32 samples = 5 value = [1, 4, 0] class = e
gini = 0.0 samples = 8 value = [8, 0, 0] class = n
Genital thrush <= 1.5 gini = 0.5 samples = 6 value = [3, 3, 0] class = n
gini = 0.0 samples = 1 value = [1, 0, 0] class = n
gini = 0.0 samples = 2 value = [0, 2, 0] class = e
gini = 0.0 samples = 4 value = [0, 4, 0] class = e
gini = 0.0 samples = 1 value = [1, 0, 0] class = n
gini = 0.0 samples = 3 value = [0, 3, 0] class = e
gini = 0.0 samples = 3 value = [3, 0, 0] class = n

Zoomed view of nodes of the tree

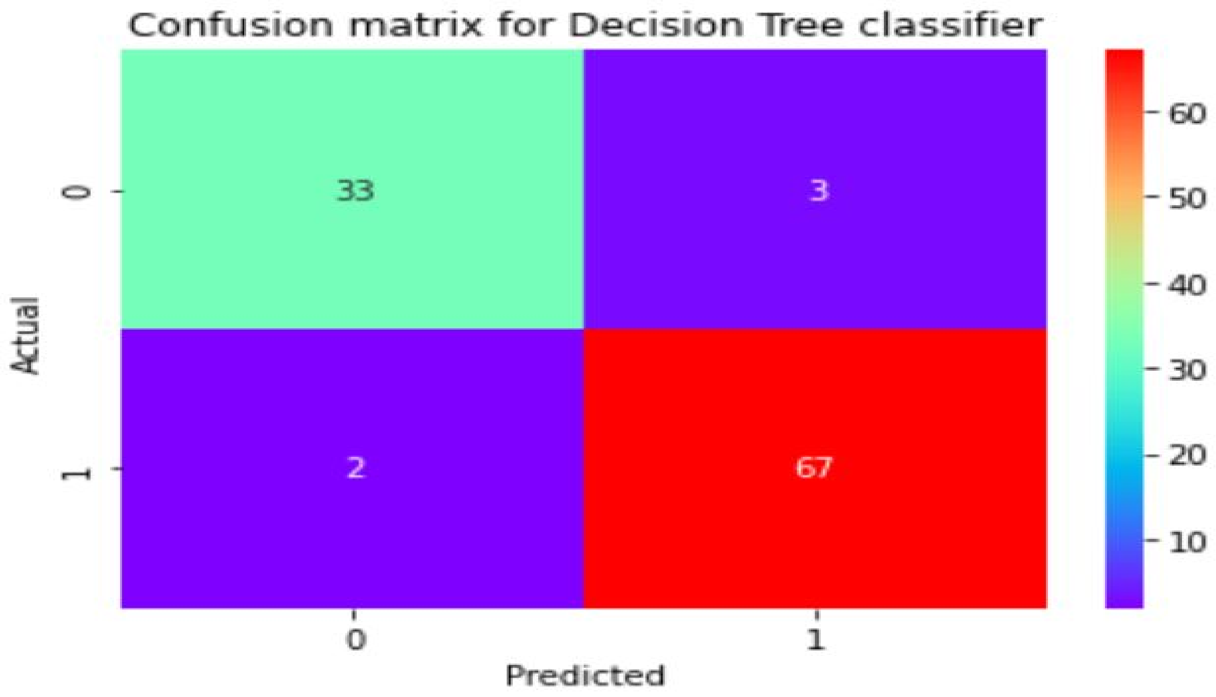


In [78]:
```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import average_precision_score
from sklearn.metrics import roc_curve, auc
```

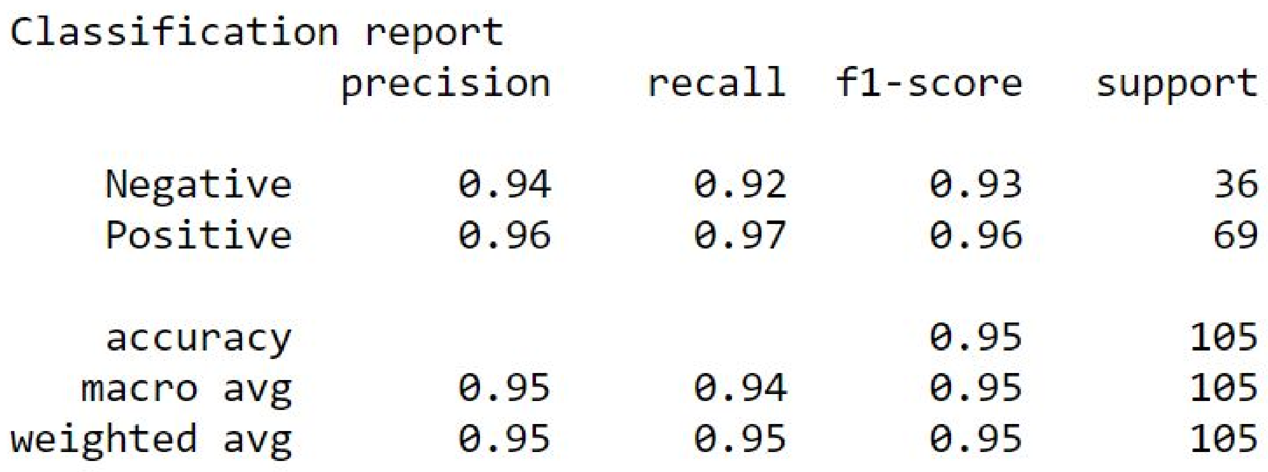
Also we can plot the precision vs recall curve for Decision Tree classifier using sklearn library by importing precision_recall_curve component and using matplot library. And we have imported certain other libraries such as roc_curve, auc, confusion_matrix and classification_report, so that we can generate the curves for all the metrics so that we can draw a comparison between the results of this and the other classification algorithm. Also we have used classification_report to generate a full fledged report of the classification.

```
In [79]: m=confusion_matrix(y_test,y_pred)
         print ('Confusion matrix plot')
         sns.heatmap(confusion_matrix(y_test,y_pred),cmap='rainbow',annot=True,fmt
         ="d")
         plt.title('Confusion matrix for Decision Tree classifier')
         plt.xlabel('Predicted')
         plt.ylabel('Actual')
         plt.show()



         print('Classification report')
         print(classification_report(y_test,y_pred))
```

```
Confusion matrix plot
```



Again we use sklearn. metrics to import the classification report component so as to calculate the precision, recall, f score and support to find how good the classifier is and to compare the gaussian naive bayes classifier with the decision tree classifier we have used.



```
Classification report
              precision    recall  f1-score   support

    Negative       0.94      0.92      0.93        36
    Positive       0.96      0.97      0.96        69

    accuracy                           0.95       105
   macro avg       0.95      0.94      0.95       105
weighted avg       0.95      0.95      0.95       105
```

Therefore we see here the precision, f score, recall, support for positive class

Precision = Positive predicate value = (TP)/(TP+FP) = **0.96**

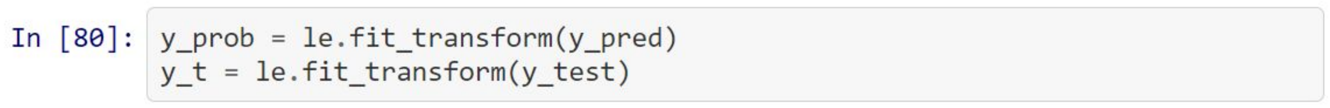
Recall = Sensitivity= TPR = **0.97**

F1- score = (2*r*p)/(r+p) = **0.96**

(r: Recall, p=Precision)

Support means that in the actual test data set there were 69 records with value as positive and 36 records as negative.

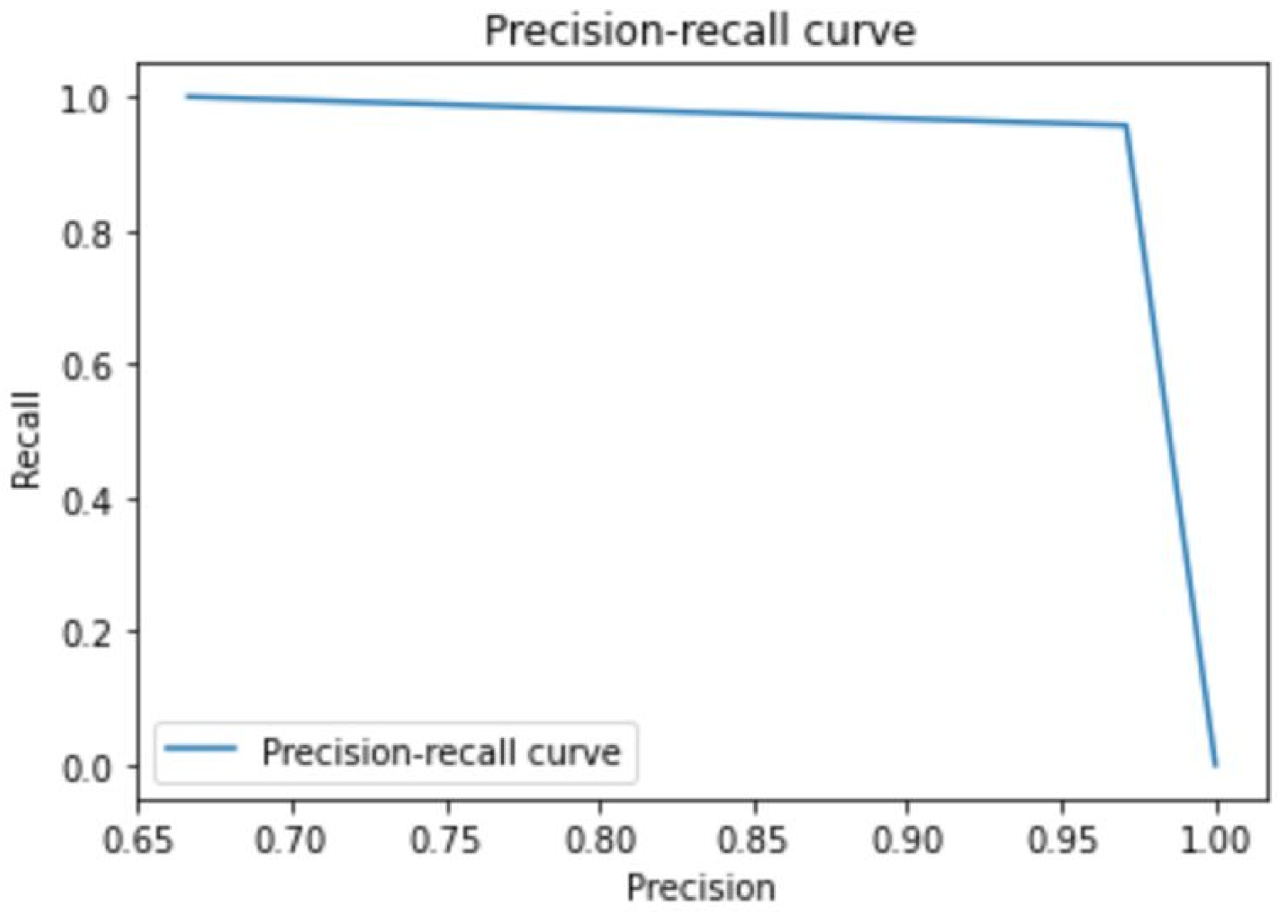
Also the value of precision, recall and f measure are near 1 which shows that the model is classifying test records quite efficiently.



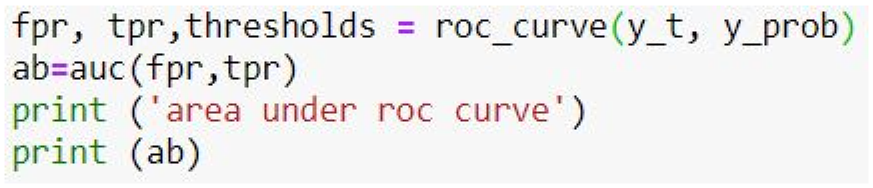
```
In [80]: y_prob = le.fit_transform(y_pred)
         y_t = le.fit_transform(y_test)
```

Transforming the predicted values again into integer from string so as to perform further operation using the fit_transform function of the label encoder available in the preprocessing library from sklearn and making it fit to be used for further processing.

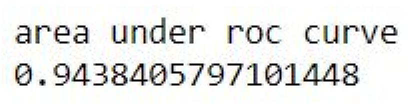
In [81]:
```
precision, recall, thresholds = precision_recall_curve(y_prob,
y_t)
plt.plot(precision, recall, label='Precision-recall curve')
plt.xlabel('Precision')
plt.ylabel('Recall')
plt.title('Precision-recall curve')
plt.legend(loc="lower left")
plt.show()
```



Precision-Recall curves summarize the trade-off between the true positive rate and the positive predictive value for a predictive model using different probability thresholds. A model with perfect skill is depicted as a point at (1,1). A skilful model is represented by a curve that bows towards (1,1) above the flat line of no skill. So we can see our model is also quite efficient. As it also plots a flat line.



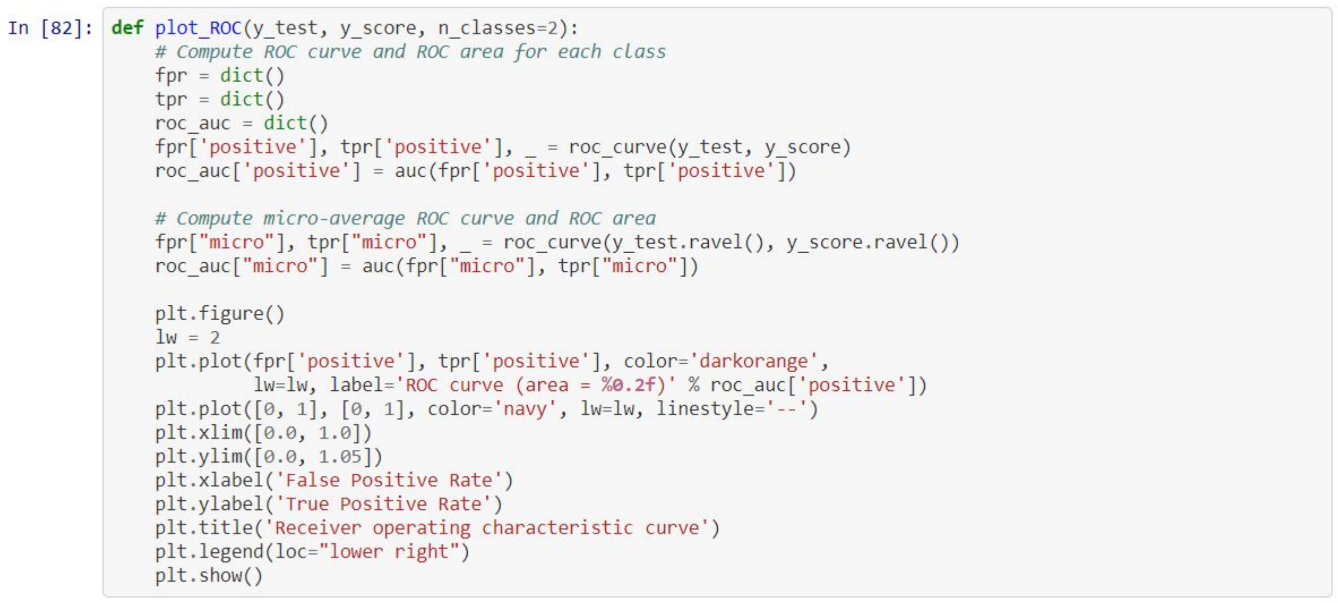
```
fpr, tpr,thresholds = roc_curve(y_t, y_prob)
ab=auc(fpr,tpr)
print ('area under roc curve')
print (ab)
```



```
area under roc curve
0.9438405797101448
```

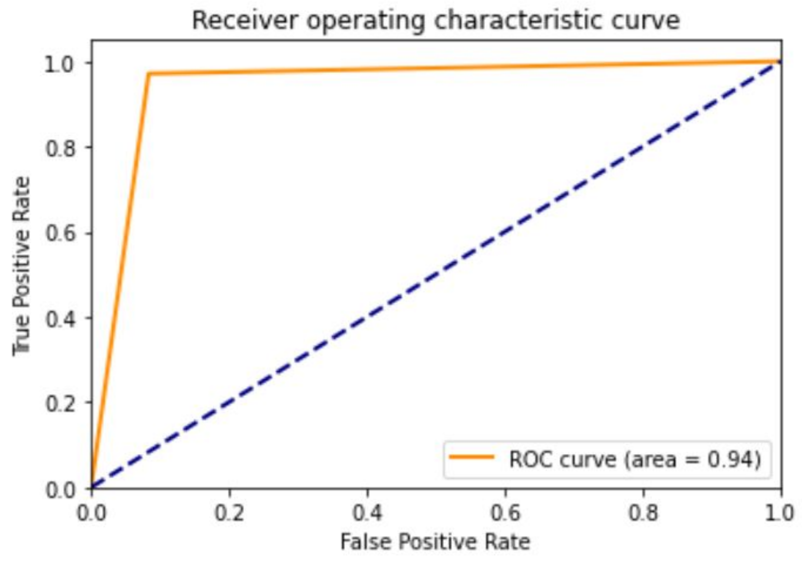
ROC (Receiver operating Characteristic) is more suitable for our data than precision recall curve to analyze our model as our data does not have a class imbalance problem.

Area under the roc curve shows how good our classifier is as the ideal area under the curve should be 1. Using the roc_curve imported from sklearn.metrics library we have calculated that the area under the curve comes out to be 0.944 ~ 0.94 which is good as it is close to 1 so our model works quite efficiently.



```
In [82]: def plot_ROC(y_test, y_score, n_classes=2):
             # Compute ROC curve and ROC area for each class
             fpr = dict()
             tpr = dict()
             roc_auc = dict()
             fpr['positive'], tpr['positive'], _ = roc_curve(y_test, y_score)
             roc_auc['positive'] = auc(fpr['positive'], tpr['positive'])

             # Compute micro-average ROC curve and ROC area
             fpr["micro"], tpr["micro"], _ = roc_curve(y_test.ravel(), y_score.ravel())
             roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])

             plt.figure()
             lw = 2
             plt.plot(fpr['positive'], tpr['positive'], color='darkorange',
                      lw=lw, label='ROC curve (area = %0.2f)' % roc_auc['positive'])
             plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
             plt.xlim([0.0, 1.0])
             plt.ylim([0.0, 1.05])
             plt.xlabel('False Positive Rate')
             plt.ylabel('True Positive Rate')
             plt.title('Receiver operating characteristic curve')
             plt.legend(loc="lower right")
             plt.show()
```

```
In [83]: plot_ROC(y_t, y_prob)
```



ROC curve plots True positive rate (TPR) on Y axis and False positive rate (FPR) on X axis and if the area under the curve is close to 1 it is an ideal condition as the ideal plot would be a straight line joining points (0,0) and (0,1). The diagonal line joins two extreme conditions of classifying all records as positive or all records as negative without classifying the records actually.

Decision Tree classifier turns out to be very efficient for our data set as the area under the roc curve i.e. 0.9 is close to 1 and can be used as our data does not have class imbalance problems.

Comparison of Gaussian Naive Bayes and Decision tree classifier

Comparison Metric	Gaussian Naive Bayes	Decision Tree
Accuracy	91.34%	95.24%
Precision	0.93	0.96
True Positive	67	67
False Positive	5	3
True Negative	4	2
False Negative	28	33

Based on the above comparison we can conclude that Decision tree classification algorithm is better than Gaussian Naive Bayes classifier because it gives better accuracy and also the false positives are less in decision tree classifier as compared to the Gaussian Naive bayes, not only this but also the precision is better in case of decision tree classifier. So, from the above comparison we can draw the conclusion that for the dataset we have used for classification, the decision tree is a better classifier as it produces more accurate and precise results.

Importance and Uses of classification for risk detection of diabetes

By classifying and predicting whether you have the risk of diabetes or not according to your age group, gender, and symptoms such as weakness, muscle stiffness, itchiness, visual blurring which a person can observe himself and does not require a medical supervision can be used to predict that a person is diabetic or not and can be useful for a person to then further seek for medical attention if classified as positive for diabetes at an early stage. A person can himself also change his dietary plans, exercise habits, sleep schedule etc. in case he is classified as positive for diabetes.

REFERENCES

1. https://seaborn.pydata.org/generated/seaborn.heatmap.html
2. https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
3. https://towardsdatascience.com/visualizing-decision-trees-with-python-scikit-learn-graph viz-matplotlib-1c50b4aa68dc
4. https://datascience.stackexchange.com/questions/9443/when-to-use-one-hot-encoding-v s-labelencoder-vs-dictvectorizor
5. https://towardsdatascience.com/understanding-the-roc-and-auc-curves-a05b68550b69
6. https://www.kaggle.com/prashant111/decision-tree-classifier-tutorial