Naive bayes

In [2]:

```python
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
data = {
    'Weather': ['Sunny', 'Sunny', 'Overcast', 'Rainy', 'Rainy', 'Rainy',
    'Temperature': ['Hot', 'Hot', 'Hot', 'Mild', 'Cool', 'Cool', 'Cool',
    'Humidity': ['High', 'High', 'High', 'High', 'Normal', 'Normal', 'Nor
    'Wind': ['Weak', 'Strong', 'Weak', 'Weak', 'Weak', 'Strong', 'Strong'
    'Play': ['No', 'No', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'No', 'Yes', '
}

df = pd.DataFrame(data)
df

```

Out[2]:

	Weather	Temperature	Humidity	Wind	Play
0	Sunny	Hot	High	Weak	No
1	Sunny	Hot	High	Strong	No
2	Overcast	Hot	High	Weak	Yes
3	Rainy	Mild	High	Weak	Yes
4	Rainy	Cool	Normal	Weak	Yes
5	Rainy	Cool	Normal	Strong	No
6	Overcast	Cool	Normal	Strong	Yes
7	Sunny	Mild	High	Weak	No
8	Sunny	Cool	Normal	Weak	Yes
9	Rainy	Mild	Normal	Weak	Yes
10	Sunny	Mild	Normal	Strong	Yes
11	Overcast	Mild	High	Strong	Yes
12	Overcast	Hot	Normal	Weak	Yes
13	Rainy	Mild	High	Strong	No

In [3]:

```
lb= LabelEncoder()
data=df.apply(lb.fit_transform)

data
```

Out[3]:

	Weather	Temperature	Humidity	Wind	Play
0	2	1	0	1	0
1	2	1	0	0	0
2	0	1	0	1	1
3	1	2	0	1	1
4	1	0	1	1	1
5	1	0	1	0	0
6	0	0	1	0	1
7	2	2	0	1	0
8	2	0	1	1	1
9	1	2	1	1	1
10	2	2	1	0	1
11	0	2	0	0	1
12	0	1	1	1	1
13	1	2	0	0	0

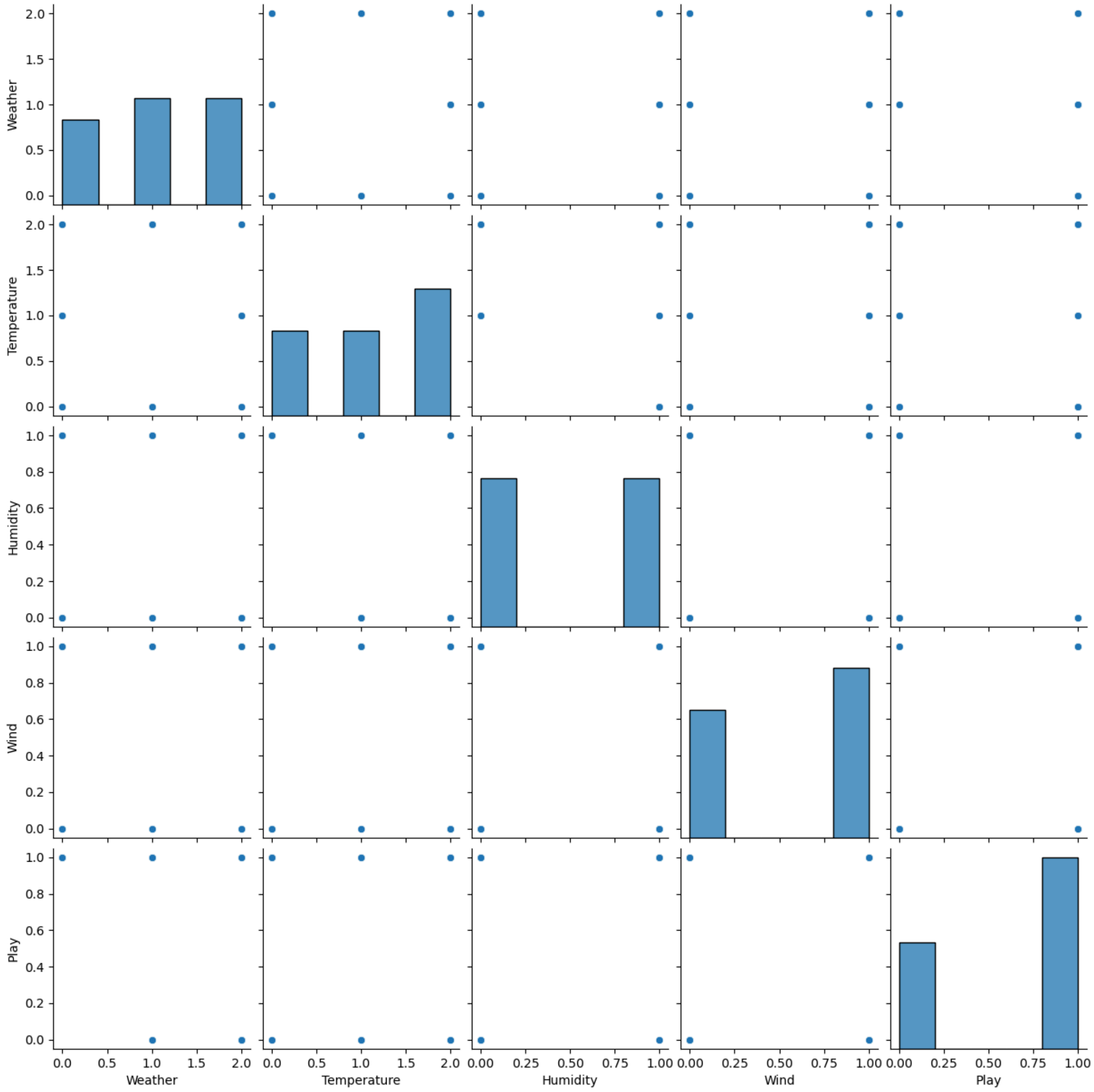
In [4]:

```
sns.pairplot(data)
```

```
C:\Users\Ahmed Islam\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: Us
erWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

Out[4]:

```
<seaborn.axisgrid.PairGrid at 0x2986c23d350>
```



In [5]:

```
df.isnull().sum()
```

Out[5]:

```
Weather        0
Temperature    0
Humidity       0
Wind           0
Play           0
dtype: int64
```

In [6]:

```
df.duplicated().sum()
```

Out[6]:

```
0
```

In [7]:

```
df.count()
```

Out[7]:

```
Weather        14
Temperature    14
Humidity       14
Wind           14
Play           14
dtype: int64
```

In [8]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14 entries, 0 to 13
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   Weather      14 non-null     object
 1   Temperature  14 non-null     object
 2   Humidity     14 non-null     object
 3   Wind         14 non-null     object
 4   Play         14 non-null     object
dtypes: object(5)
memory usage: 692.0+ bytes
```

In [9]:

```
df.describe()
```

Out[9]:

	Weather	Temperature	Humidity	Wind	Play
count	14	14	14	14	14
unique	3	3	2	2	2
top	Sunny	Mild	High	Weak	Yes
freq	5	6	7	8	9

In [10]:

```
df.nunique()
```

Out[10]:

```
Weather        3
Temperature    3
Humidity       2
Wind           2
Play           2
dtype: int64
```

In [11]:

```
df['Play']
```

Out[11]:

```
0      No
1      No
2     Yes
3     Yes
4     Yes
5      No
6     Yes
7      No
8     Yes
9     Yes
10    Yes
11    Yes
12    Yes
13     No
Name: Play, dtype: object
```

In [12]:

```
len(df['Play'])
```

Out[12]:

```
14
```

In [13]:

```
count_yes=len(df[df['Play']=='Yes'])
count_no=len(df[df['Play']=='No'])
print(count_yes)
print(count_no)
```

```
9
5
```

In [14]:

```
target = 'Play'
def calculate_prior(df, target):
    total_count = len(df)
    count_yes = len(df[df[target] == 'Yes'])
    count_no = len(df[df[target] == 'No'])

    p_yes = count_yes / total_count
    p_no = count_no / total_count

    print("probability of yes is:" ,p_yes)
    print("probability of no is:" ,p_no)
calculate_prior(df,target)

```

```
probability of yes is: 0.6428571428571429
probability of no is: 0.35714285714285715
```

In [15]:

```
features =   ['Weather', 'Temperature', 'Humidity', 'Wind']
test_sample =[ 'Sunny',     'Cool',       'Low',    'Weak']
target= 'Play'
```

In [16]:

```
sunny_rows=len(df[df['Weather']=='Sunny'])
sunny_rows
```

Out[16]:

```
5
```

In [17]:

```

sunny_rows = df[df['Weather'] == 'Sunny']

sunny_yes = len(sunny_rows[sunny_rows['Play'] == 'Yes'])
sunny_no = len(sunny_rows[sunny_rows['Play'] == 'No'])


print("Sunny_yes:", sunny_yes)
print("Sunny_no:", sunny_no)

print(sunny_yes/count_yes)

print(sunny_no/count_no)

```

```
Sunny_yes: 2
Sunny_no: 3
0.2222222222222222
0.6
```

In [18]:

```
cool_rows=len(df[df['Temperature']=='Cool'])
cool_rows
```

Out[18]:

```
4
```

In [19]:

```

cool_rows = df[df['Temperature'] == 'Cool']

cool_yes = len(cool_rows[cool_rows['Play'] == 'Yes'])
cool_no = len(cool_rows[cool_rows['Play'] == 'No'])

print("cool_yes:", cool_yes)
print("cool_no:", cool_no)

print(cool_yes/count_yes)

print(cool_no/count_no)
```

```
cool_yes: 3
cool_no: 1
0.3333333333333333
0.2
```

In [20]:

```
humidity_rows=len(df[df['Humidity']=='Low'])
humidity_rows
```

Out[20]:

```
0
```

In [21]:

```python


humidity_rows = df[df['Humidity'] == 'High']

humidity_yes = len(humidity_rows[humidity_rows['Play'] == 'Yes'])
humidity_no = len(humidity_rows[humidity_rows['Play'] == 'No'])

# Print results
print("humidity_yes:", humidity_yes)
print("humidity_no:", humidity_no)

print(humidity_yes/count_yes)

print(humidity_no/count_no)
```

```
humidity_yes: 3
humidity_no: 4
0.3333333333333333
0.8
```

In [22]:

```python
wind_rows=len(df[df['Wind']=='Weak'])

wind_rows
```

Out[22]:

```
8
```

In [23]:

```python

wind_rows = df[df['Wind'] == 'Weak']

wind_yes = len(wind_rows[wind_rows['Play'] == 'Yes'])
wind_no = len(wind_rows[wind_rows['Play'] == 'No'])

print("count_yes:", wind_yes)
print("count_no:", wind_no)

print(wind_yes/count_yes)

print(wind_no/count_no)

```

```
count_yes: 6
count_no: 2
0.6666666666666666
0.4
```

In [24]:

```python
import numpy as np
import pandas as pd


test_sample =[ 'Sunny',     'Cool',       'Low',    'Weak']
target='Play'



total_count = len(target)
print("Total number of samples:",total_count)


total_yes = len(df[df['Play'] == 'Yes'])
total_no = len(df[df['Play'] ==  'No'])

print("Total number of 'Yes':", total_yes)
print("Total number of 'No':",  total_no)

def calculate_prior(df, target):
    total_count = len(df[target])
    count_yes = len(df[df[target] == 'Yes'])
    count_no = len(df[df[target] ==  'No'])

    p_yes = count_yes / total_count
    p_no = count_no / total_count

    return p_yes, p_no

p_yes, p_no = calculate_prior(df, target)
print("Probability of 'Yes':", p_yes)
print("Probability of 'No':", p_no)


sunny_rows = df[df['Weather'] == 'Sunny']
sunny_yes = len(sunny_rows[sunny_rows['Play'] == 'Yes'])
sunny_no = len(sunny_rows[sunny_rows['Play'] == 'No'])

print("Sunny and 'Yes':", sunny_yes)
print("Sunny and 'No':", sunny_no)

cool_rows = df[df['Temperature'] == 'Cool']
cool_yes = len(cool_rows[cool_rows['Play'] == 'Yes'])
cool_no = len(cool_rows[cool_rows['Play'] == 'No'])

print("Cool and 'Yes':", cool_yes)
print("Cool and 'No':", cool_no)

humidity_rows = df[df['Humidity'] == 'High']
humidity_yes = len(humidity_rows[humidity_rows['Play'] == 'Yes'])
humidity_no = len(humidity_rows[humidity_rows['Play'] == 'No'])

print("Humidity 'High' and 'Yes':", humidity_yes)
print("Humidity 'High' and 'No':", humidity_no)

wind_rows = df[df['Wind'] == 'Weak']
wind_yes = len(wind_rows[wind_rows['Play'] == 'Yes'])
```

```python
wind_no = len(wind_rows[wind_rows['Play'] == 'No'])

print("Wind 'Weak' and 'Yes':", wind_yes)
print("Wind 'Weak' and 'No':", wind_no)

# Calculate feature probabilities
p_sunny_yes = sunny_yes / total_yes
p_sunny_no = sunny_no / total_no

p_cool_yes = cool_yes / total_yes
p_cool_no = cool_no / total_no

p_humidity_high_yes = humidity_yes / total_yes
p_humidity_high_no = humidity_no / total_no

p_wind_yes = wind_yes / total_yes
p_wind_no = wind_no / total_no

# Calculate probabilities for the test sample
prob_sunny_cool_high_weak_yes = p_sunny_yes * p_cool_yes * p_humidity_hig
prob_sunny_cool_high_weak_no = p_sunny_no * p_cool_no * p_humidity_high_n

print("Probability of 'Yes' given the features:", prob_sunny_cool_high_we
print("Probability of 'No' given the features:", prob_sunny_cool_high_wea

# Determine the predicted class
if prob_sunny_cool_high_weak_yes > prob_sunny_cool_high_weak_no:
    print("The predicted class is: Yes")
else:
    print("The predicted class is: No")

```

```
Total number of samples: 4
Total number of 'Yes': 9
Total number of 'No': 5
Probability of 'Yes': 0.6428571428571429
Probability of 'No': 0.35714285714285715
Sunny and 'Yes': 2
Sunny and 'No': 3
Cool and 'Yes': 3
Cool and 'No': 1
Humidity 'High' and 'Yes': 3
Humidity 'High' and 'No': 4
Wind 'Weak' and 'Yes': 6
Wind 'Weak' and 'No': 2
Probability of 'Yes' given the features: 0.010582010582010581
Probability of 'No' given the features: 0.013714285714285715
The predicted class is: No
```

with Dataset diabetes classification data....

In [25]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

data = pd.read_csv("Naive-Bayes-Classification-Data.csv")
data
```

Out[25]:

	glucose	bloodpressure	diabetes
0	40	85	0
1	40	92	0
2	45	63	1
3	45	80	0
4	40	73	1
...	...	...	...
990	45	87	0
991	40	83	0
992	40	83	0
993	40	60	1
994	45	82	0

995 rows × 3 columns

In [26]:

```
data.isnull().sum()
```

Out[26]:

```
glucose          0
bloodpressure    0
diabetes         0
dtype: int64
```

In [27]:

```
data.duplicated().sum()
```

Out[27]:

```
820
```

In [28]:

```
dataset=data.drop_duplicates()
dataset
```

Out[28]:

	glucose	bloodpressure	diabetes
0	40	85	0
1	40	92	0
2	45	63	1
3	45	80	0
4	40	73	1
...	...	...	...
873	20	73	1
914	55	87	0
953	40	75	0
955	45	50	1
979	50	83	1

175 rows × 3 columns

In [29]:

```
dataset.shape
```

Out[29]:

```
(175, 3)
```

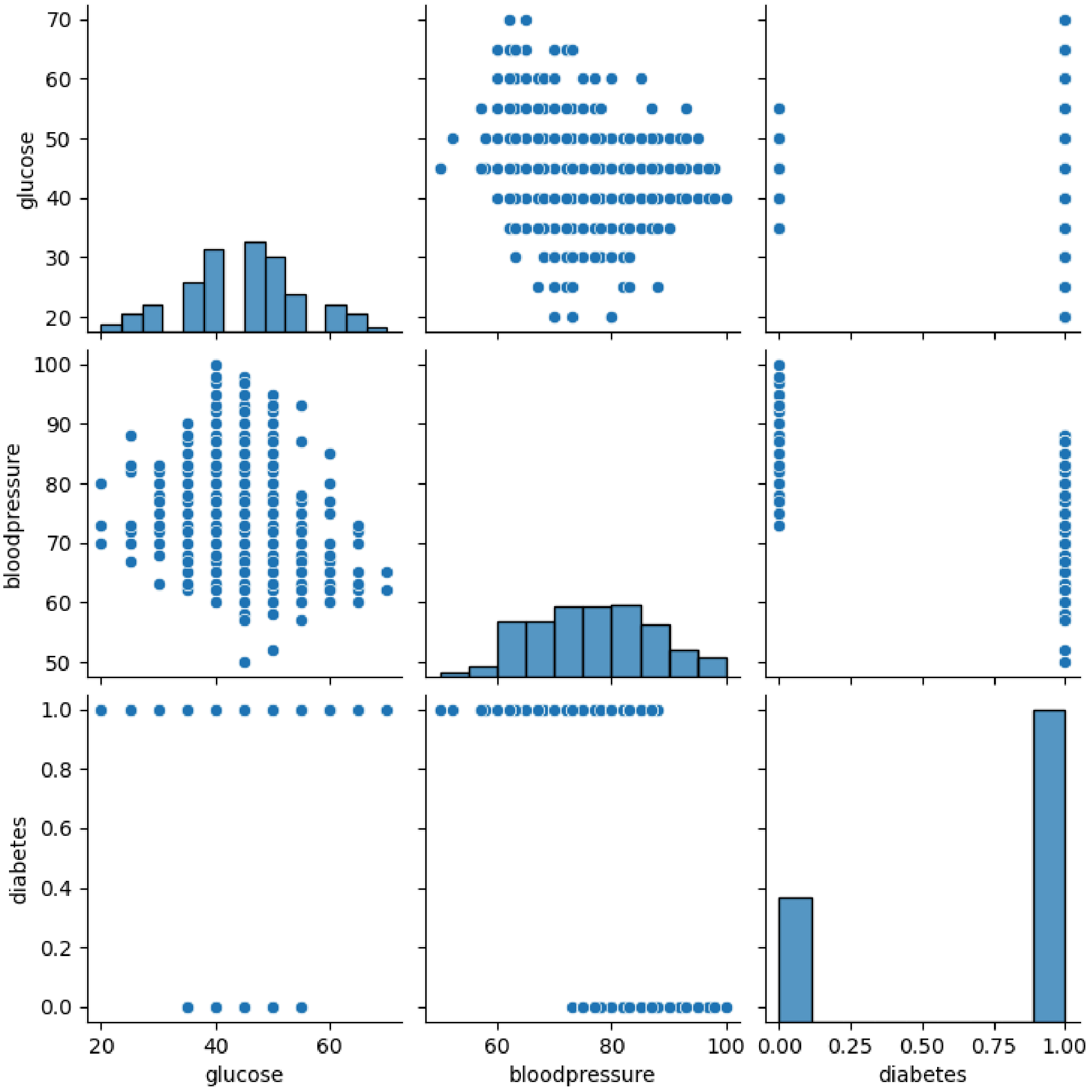
In [30]:

```
sns.pairplot(dataset)
```

```
C:\Users\Ahmed Islam\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: Us
erWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

Out[30]:

```
<seaborn.axisgrid.PairGrid at 0x29873677290>
```



In [31]:

```
dataset.count()
```

Out[31]:

```
glucose          175
bloodpressure    175
diabetes         175
dtype: int64
```

In [32]:

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 175 entries, 0 to 979
Data columns (total 3 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   glucose        175 non-null    int64
 1   bloodpressure  175 non-null    int64
 2   diabetes       175 non-null    int64
dtypes: int64(3)
memory usage: 5.5 KB
```

In [33]:

```
dataset.describe()
```

Out[33]:

	glucose	bloodpressure	diabetes
count	175.000000	175.000000	175.000000
mean	44.485714	75.874286	0.714286
std	10.367864	10.515788	0.453050
min	20.000000	50.000000	0.000000
25%	40.000000	68.000000	0.000000
50%	45.000000	77.000000	1.000000
75%	50.000000	83.000000	1.000000
max	70.000000	100.000000	1.000000

In [34]:

```
dataset.nunique()
```

Out[34]:

```
glucose          11
bloodpressure    29
diabetes          2
dtype: int64
```

In [40]:

```python
test_sample = {'glucose':40, 'bloodpressure':83}
target = 'diabetes'

total_0 = len(dataset[dataset[target] == 0])
total_1 = len(dataset[dataset[target] == 1])

def prior(dataset, target):
    total_count = len(dataset[target])
    total_0 = len(dataset[dataset[target] == 0])
    total_1 = len(dataset[dataset[target] == 1])

    prob_0 = total_0 / total_count
    prob_1 = total_1 / total_count

    return prob_0, prob_1


prob_0, prob_1 = prior(dataset, target)

print("Prior Probability of class 0:", prob_0)
print("Prior Probability of class 1:", prob_1)


```

```
Prior Probability of class 0: 0.2857142857142857
Prior Probability of class 1: 0.7142857142857143
```

In [41]:

```

row_43 = dataset[dataset["glucose"] == 40]


total_43_0 = len(row_43[row_43['diabetes'] == 0])
print("Total for 0 (glucose):", total_43_0)

total_43_1 = len(row_43[row_43['diabetes'] == 1])
print("Total for 1 (glucose):", total_43_1)

#conditional probabilities
prob_43_0 = total_43_0 / total_0
print(prob_43_0)

prob_43_1 = total_43_1 / total_1
print(prob_43_1)


row_83 = data[data["bloodpressure"] == 83]

total_83_0 = len(row_83[row_83['diabetes'] == 0])
print("Total for 0 (bloodpressure):", total_83_0)

total_83_1 = len(row_83[row_83['diabetes'] == 1])
print("Total for 1 (bloodpressure):", total_83_1)

#conditional probabilities
prob_83_0 = total_83_0 / total_0
print(prob_83_0)

prob_83_1 = total_83_1 / total_1
print(prob_83_1)

#final probabilities
final_prob_1 = prob_43_1 * prob_83_1 * prob_1
print("This is the final probability for (glucose(1),bloodpressure(1)):",

final_prob_0 = prob_43_0 * prob_83_0 * prob_0
print("This is the final probability for (glucose(0)),bloodpressure(0)):"


if final_prob_0 > final_prob_1:
    print("The new point belongs to class 0 (no diabetes)")
else:
    print("The new point belongs to class 1")

Total for 0 (glucose): 16
Total for 1 (glucose): 17
0.32
0.136
Total for 0 (bloodpressure): 57
Total for 1 (bloodpressure): 9
1.14
0.072
This is the final probability for (glucose(1),bloodpressure(1)): 0.006994285
714285715
This is the final probability for (glucose(0)),bloodpressure(0)): 0.10422857
14285714
The new point belongs to class 0 (no diabetes)
```

In [48]:

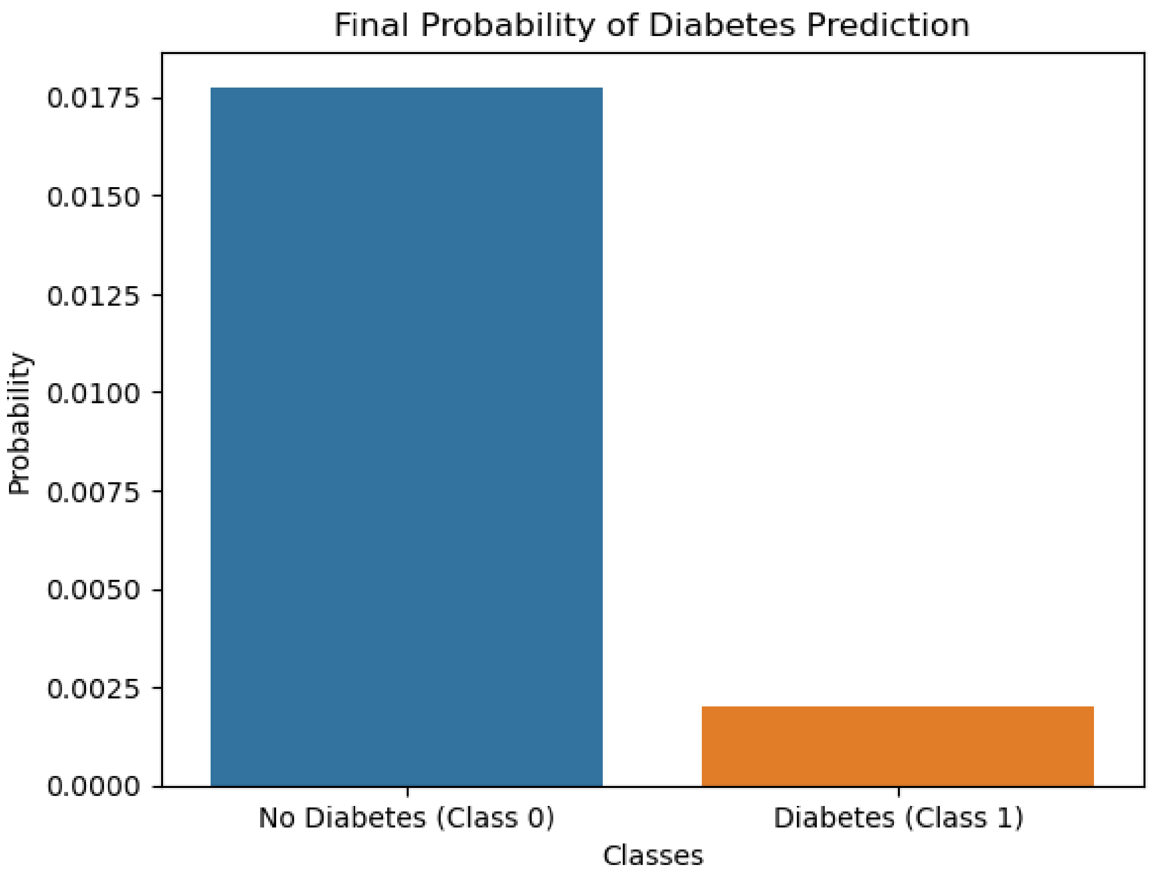
```python
probabilities = [final_prob_0, final_prob_1]
classes = ['No Diabetes (Class 0)', 'Diabetes (Class 1)']

sns.barplot(x=classes, y=probabilities)

plt.xlabel('Classes')
plt.ylabel('Probability')
plt.title('Final Probability of Diabetes Prediction')

plt.show()

```



In [46]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

dataset= pd.read_csv('Naive-Bayes-Classification-Data.csv')
#calculate prior probabilities
total_0 = len(dataset[dataset[target] == 0])
total_1 = len(dataset[dataset[target] == 1])

def prior(dataset, target):
    total_count = len(dataset[target])
    total_0 = len(dataset[dataset[target] == 0])
    total_1 = len(dataset[dataset[target] == 1])

    prob_0 = total_0 / total_count
    prob_1 = total_1 / total_count

    return prob_0, prob_1, total_0, total_1


prob_0, prob_1, total_0, total_1 = prior(dataset, target)

print("Prior Probability of class 0 (No diabetes):", prob_0)
print("Prior Probability of class 1 (Diabetes):", prob_1)

# Function to calculate conditional probabilities
def conditional_prob(dataset, feature, value, target, total_0, total_1):

    feature_data = dataset[dataset[feature] == value]

    # Calculate the conditional probabilities for both classes
    total_feature_0 = len(feature_data[feature_data[target] == 0])
    total_feature_1 = len(feature_data[feature_data[target] == 1])


    prob_feature_0 = (total_feature_0) / (total_0 )
    prob_feature_1 = (total_feature_1) / (total_1)

    return prob_feature_0, prob_feature_1

# Conditional probabilities for 'glucose' and 'bloodpressure'
prob_glucose_0, prob_glucose_1 = conditional_prob(dataset, 'glucose', tes
prob_bloodpressure_0, prob_bloodpressure_1 = conditional_prob(dataset, 'b

# final probability
final_prob_0 = prob_glucose_0 * prob_bloodpressure_0 * prob_0
final_prob_1 = prob_glucose_1 * prob_bloodpressure_1 * prob_1


print(f"Final Probability of class 0 (No diabetes): {final_prob_0}")
print(f"Final Probability of class 1 (Diabetes): {final_prob_1}")

if final_prob_0 > final_prob_1:
    print("The new point belongs to class 0 (No diabetes)")
else:
    print("The new point belongs to class 1 (Diabetes)")
```

```
Prior Probability of class 0 (No diabetes): 0.49949748743718594
Prior Probability of class 1 (Diabetes): 0.5005025125628141
Final Probability of class 0 (No diabetes): 0.0177507254582773
Final Probability of class 1 (Diabetes): 0.0019979415148029304
The new point belongs to class 0 (No diabetes)
```

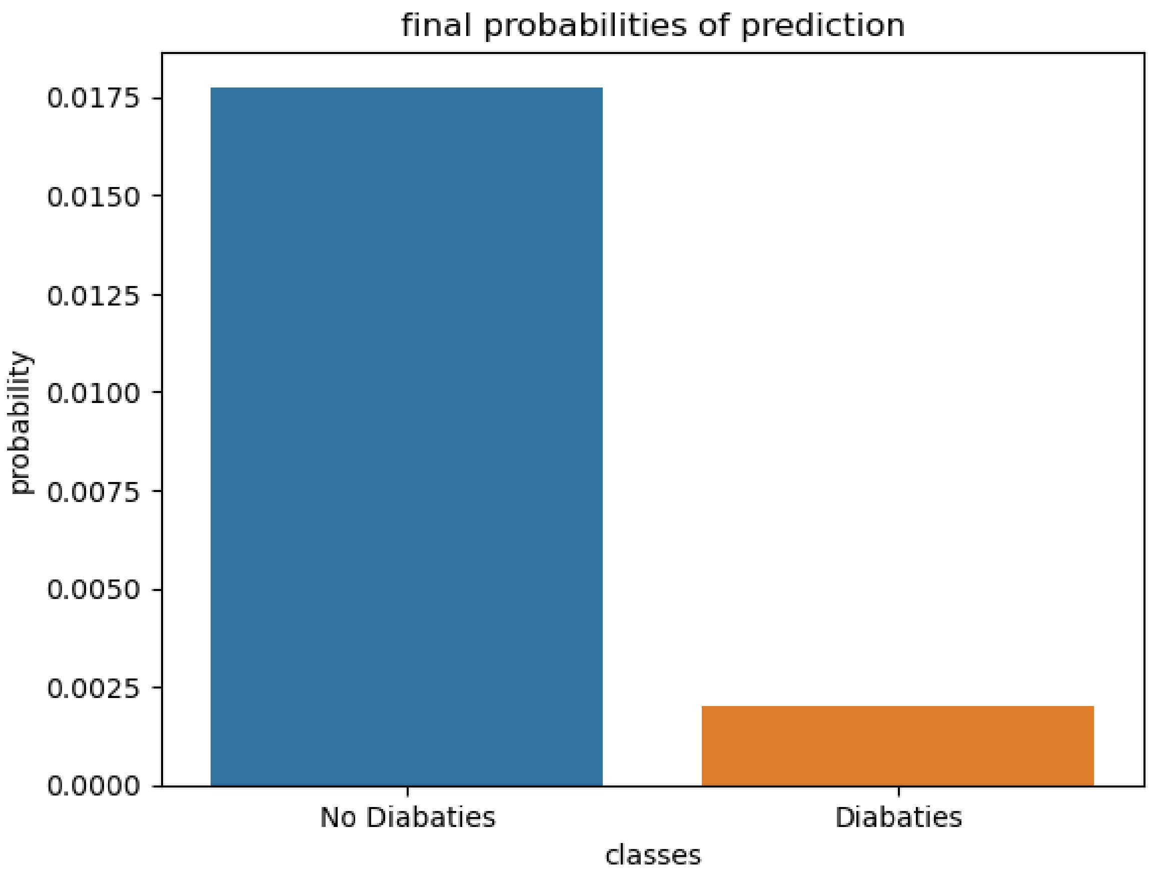
In [61]:

```python
import seaborn as sns
import matplotlib.pyplot as plt

probabilities= [final_prob_0, final_prob_1]
classes=['No Diabaties','Diabaties']

sns.barplot(x=classes, y=probabilities)
plt.title('final probabilities of prediction')
plt.xlabel('classes')
plt.ylabel('probability')
plt.show()

```



In [ ]:

```python

```