DIABETES DIAGNOSIS ANALYSIS

19TH JUNE,2023

1 Problem Statement : Diabetes Diagnosis Analysis

1.1 Description:

> Diabetes Mellitus is one of the currently deadliest diseases as it set the foundation for several conditions to develop in the body. Diabetes paves way for conditions such as high blood pressure, hyperlipidaemia, neuropathy, eye problems, kidney failure, among others. All these conditions also lead to other conditions that are also very dangerous to the body, making diabetes worth studied into more details. Between 2000 and 2019, there was a 3% increase in diabetes mortality rates by age. In 2019, diabetes and kidney disease due to diabetes caused an estimated 2 million deaths. In the diagnosis of diabetes, an important biomarker is blood glucose level, other biomarkers and physical characteristics play important role in the diagnosis as well. This article seeks the investigate the various factors involved in diabetes diagnosis to reveal new other more important features that can be used to detect the presence of diabetes and also to develop a Machine Learning Model to determine the diabetic state of a patient when their information is inputed.

2 1. Importing Libraries

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import mean_absolute_error
import warnings
warnings.filterwarnings('ignore')
```

2 The DataSets

2.1 DataSets Information

Column attribute Description Patient number Identifies patients by number Cholesterol Total cholesterol Glucose Fasting blood sugar HDL HDL or good cholesterol Chol/HDL Ratio of total cholesterol to good cholesterol. Desirable result is < 5 Age All adult African Americans

Gender 162 males, 228 females Height In inches Weight In pounds (lbs) BMI 703 x weight (lbs)/ [height(inches]2 Systolic BP The upper number of blood pressure Diastolic BP The lower number of blood pressure Waist Measured in inches Hip Measured in inches Waist/hip Ratio is possibly a stronger risk factor for heart disease than BMI Diabetes Yes (60), No (330)

This dataset is a modified data from VanderBilt but the original dataset is from a study of rural African American in Virginia. The dataset already has 13 patients dropped due to missing data. It also has two features added, being Chol/HDL and Waist/hip.

2.2 Reading Datasets

In [2]:

```
df = pd.read_csv('\\Users\\Daniel Appiah\\Desktop\\Diabetes_Classification.c
df.head()
```

Out[2]:

Patient number	Cholesterol	Glucose	HDL Chol	Chol/HDL ratio	Age	Gender	Height	Weight	BMI	Systolic BP	Diastol B
1	193	77	49	3.9	19	female	61	119	22.5	118	7
2	146	79	41	3.6	19	female	60	135	26.4	108	5
3	217	75	54	4.0	20	female	67	187	29.3	110	7
4	226	97	70	3.2	20	female	64	114	19.6	122	6
5	164	91	67	2.4	20	female	70	141	20.2	122	8

2.3 Data Exploration

In [3]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 390 entries, 1 to 390
Data columns (total 17 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Cholesterol      390 non-null    int64  
 1   Glucose          390 non-null    int64  
 2   HDL Chol         390 non-null    int64  
 3   Chol/HDL ratio   390 non-null    float64
 4   Age              390 non-null    int64  
 5   Gender           390 non-null    object 
 6   Height           390 non-null    int64  
 7   Weight           390 non-null    int64  
 8   BMI              390 non-null    float64
 9   Systolic BP      390 non-null    int64  
 10  Diastolic BP     390 non-null    int64  
 11  waist            390 non-null    int64  
 12  hip              390 non-null    int64  
 13  Waist/hip ratio  390 non-null    float64
 14  Diabetes         390 non-null    object 
 15  Unnamed: 16      1 non-null      float64
 16  Unnamed: 17      1 non-null      float64
dtypes: float64(5), int64(10), object(2)
memory usage: 54.8+ KB
```

All features here are numeric except the column named Diabetes.

In [4]:
```
df.describe()
```

Out[4]:

	Cholesterol	Glucose	HDL Chol	Chol/HDL ratio	Age	Height	Weight	
count	390.000000	390.000000	390.000000	390.000000	390.000000	390.000000	390.000000	390
mean	207.230769	107.338462	50.266667	4.524615	46.774359	65.951282	177.407692	28
std	44.666005	53.798188	17.279069	1.736634	16.435911	3.918867	40.407824	6
min	78.000000	48.000000	12.000000	1.500000	19.000000	52.000000	99.000000	15
25%	179.000000	81.000000	38.000000	3.200000	34.000000	63.000000	150.250000	24
50%	203.000000	90.000000	46.000000	4.200000	44.500000	66.000000	173.000000	27
75%	229.000000	107.750000	59.000000	5.400000	60.000000	69.000000	200.000000	32
max	443.000000	385.000000	120.000000	19.300000	92.000000	76.000000	325.000000	55

In [5]:
```
df.corr()
```

Out[5]:

	Cholesterol	Glucose	HDL Chol	Chol/HDL ratio	Age	Height	Weight	BM
Cholesterol	1.000000	0.158102	0.193162	0.475927	0.247333	-0.063601	0.062359	0.09169
Glucose	0.158102	1.000000	-0.158302	0.282210	0.294392	0.098052	0.190358	0.12928
HDL Chol	0.193162	-0.158302	1.000000	-0.681867	0.028210	-0.087238	-0.291883	-0.24186
Chol/HDL ratio	0.475927	0.282210	-0.681867	1.000000	0.163201	0.081162	0.278812	0.22840
Age	0.247333	0.294392	0.028210	0.163201	1.000000	-0.082229	-0.056784	-0.00916
Height	-0.063601	0.098052	-0.087238	0.081162	-0.082229	1.000000	0.255389	-0.25958
Weight	0.062359	0.190358	-0.291883	0.278812	-0.056784	0.255389	1.000000	0.86014
BMI	0.091695	0.129286	-0.241860	0.228407	-0.009164	-0.259589	0.860147	1.00000
Systolic BP	0.207741	0.162777	0.031807	0.115505	0.453417	-0.040704	0.097497	0.12140
Diastolic BP	0.166241	0.020262	0.078342	0.038242	0.068649	0.043617	0.166477	0.14530
waist	0.134038	0.222336	-0.276697	0.313262	0.150585	0.057447	0.847766	0.81070
hip	0.093364	0.138223	-0.223837	0.208902	0.004675	-0.095906	0.826985	0.88172
Waist/hip ratio	0.091847	0.185117	-0.158777	0.243329	0.275188	0.252548	0.250461	0.10087
Unnamed: 16	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
Unnamed: 17	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na

2.4 Handling Missing Data

In [6]:

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

Out[6]:

```
Cholesterol          0
Glucose              0
HDL Chol             0
Chol/HDL ratio       0
Age                  0
Gender               0
Height               0
Weight               0
BMI                  0
Systolic BP          0
Diastolic BP         0
waist                0
hip                  0
Waist/hip ratio      0
Diabetes             0
Unnamed: 16        389
Unnamed: 17        389
dtype: int64
```

The data has no missing values except for the last 2 unnamed features, which will be dropped due to the large number of null values.

In [7]:
```
df= df.drop(['Unnamed: 16','Unnamed: 17'], axis=1)
df.columns
```

Out[7]:
```
Index(['Cholesterol', 'Glucose', 'HDL Chol', 'Chol/HDL ratio', 'Age', 'Gende
r',
       'Height', 'Weight', 'BMI', 'Systolic BP', 'Diastolic BP', 'waist',
       'hip', 'Waist/hip ratio', 'Diabetes'],
      dtype='object')
```

2.5 Categorical Features

In [8]:
```
unique_values= df.apply(lambda x:x.nunique())
unique_values
```

Out[8]:
```
Cholesterol        153
Glucose            116
HDL Chol            75
Chol/HDL ratio      69
Age                 68
Gender               2
Height              22
Weight             139
BMI                193
Systolic BP         71
Diastolic BP        56
waist               30
hip                 32
Waist/hip ratio     39
Diabetes             2
dtype: int64
```

From this and a critical look at the data, the 'Gender' and 'Diabetes' columns are nominal features and must be changed to categorical data type.

In [9]:
```
df['Gender'] =df['Gender'].astype('category')
df['Diabetes'] =df['Diabetes'].astype('category')
df['Diabetes'].dtypes
```

Out[9]:
```
CategoricalDtype(categories=['Diabetes', 'No diabetes'], ordered=False)
```

3. Data Visualization

3.1 Visualising the distribution of the data

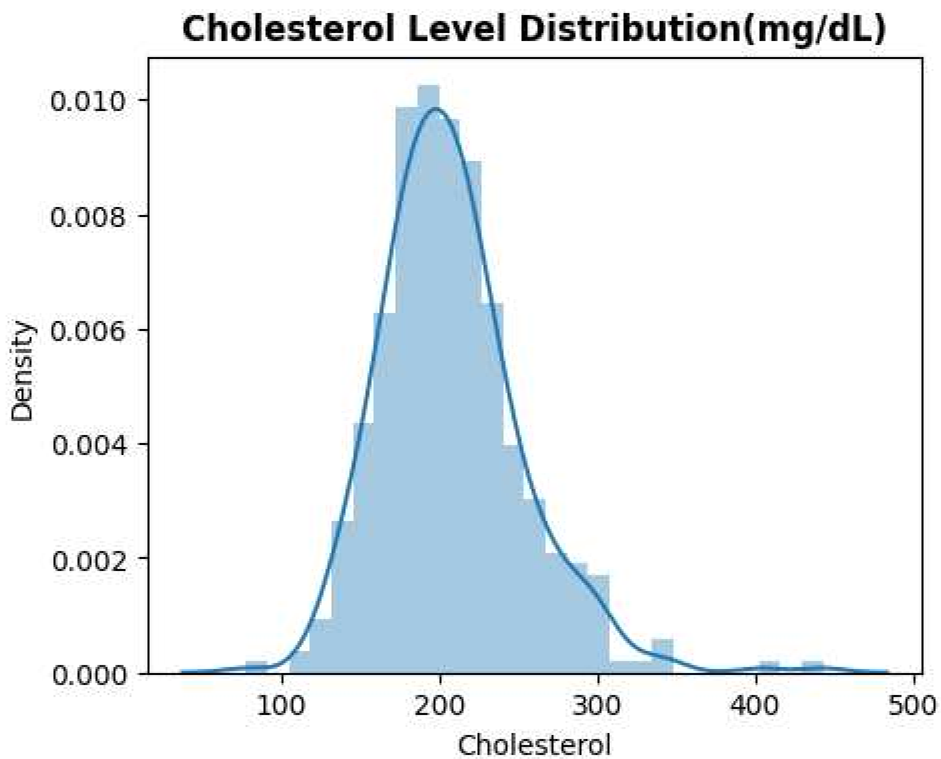
Cholesterol level

In [10]:

```
plt.figure(figsize=(5,4))
sns.distplot(df.Cholesterol)
plt.title('Cholesterol Level Distribution(mg/dL)', fontdict={'fontweight': '
#Classifying the cholesterol level.
desirable = df[df.Cholesterol < 200].count()[0]
borderline = df[(df.Cholesterol >= 200) & (df.Cholesterol < 240)].count()[0]
high = df[df.Cholesterol >= 240].count()[0]

print(f'Desirable Range (below 200 mg/dL): {desirable} patients.')
print(f'Borderline High Range (200-239 mg/dL): {borderline} patients.')
print(f'High Range (240 mg/dL and above): {high} patients.')
```

```
Desirable Range (below 200 mg/dL): 184 patients.
Borderline High Range (200-239 mg/dL): 130 patients.
High Range (240 mg/dL and above): 76 patients.
```



Glucose level

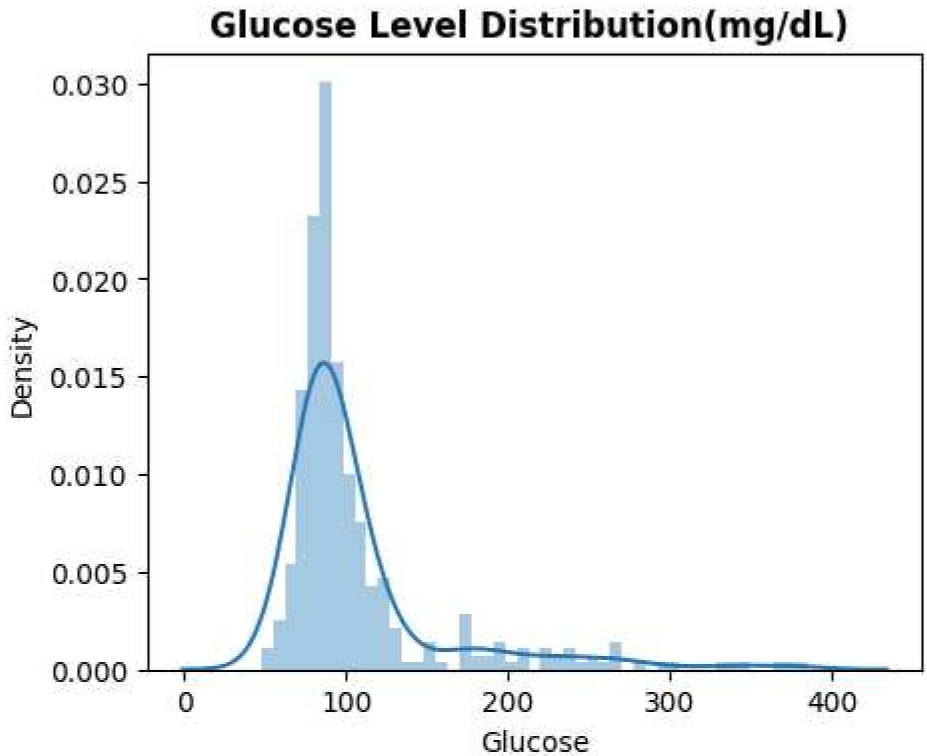
In [11]:

```
plt.figure(figsize=(5,4))
sns.distplot(df.Glucose)
plt.title('Glucose Level Distribution(mg/dL) ',fontdict={'fontweight': 'bold

desirable = df[df.Glucose < 100].count()[0]
borderline = df[(df.Glucose >= 100) & (df.Glucose < 125)].count()[0]
diabetes = df[df.Glucose >= 126].count()[0]

print(f'Normal Range (below 100 mg/dL): {desirable} patients.')
print(f'Prediabetes Range (100-125 mg/dL): {borderline} patients.')
print(f'Diabetes Range (126 mg/dL and above): {diabetes} patients.')
```

```
Normal Range (below 100 mg/dL): 260 patients.
Prediabetes Range (100-125 mg/dL): 69 patients.
Diabetes Range (126 mg/dL and above): 60 patients.
```



High density Lipoprotein

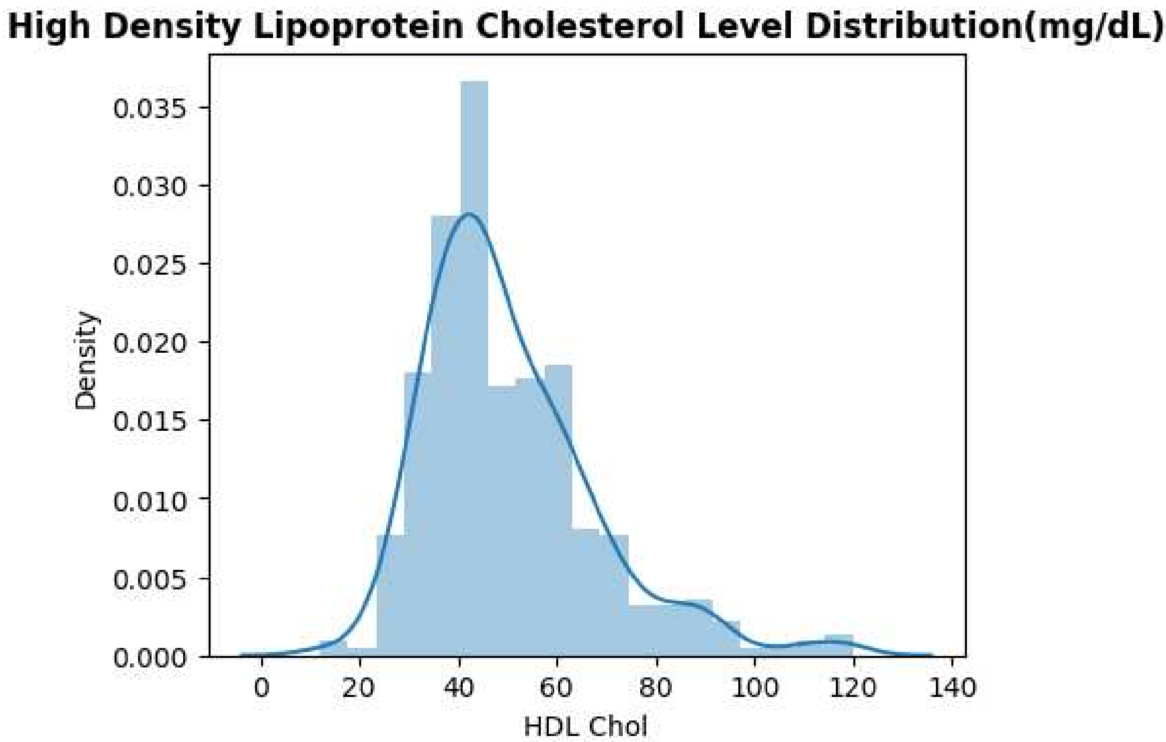
In [12]:

```python
plt.figure(figsize=(5,4))
sns.distplot(df['HDL Chol'])
plt.title('High Density Lipoprotein Cholesterol Level Distribution(mg/dL)',

desirable_men = df[(df['HDL Chol'] >= 40) & (df['HDL Chol'] < 60) ].count()[
desirable_women = df[(df['HDL Chol'] >= 50) & (df['HDL Chol'] < 60)].count()
low_men = df[df['HDL Chol'] <40 ].count()[0]
low_women= df[df['HDL Chol'] <50 ].count()[0]
high = df[df['HDL Chol'] >= 60].count()[0]

print(f'Low Range for men (below 40 mg/dL): {low_men} patients.')
print(f'Low Range for women (below 50 mg/dL): {low_women} patients.')
print(f'Normal HDL Range for men (40-59 mg/dL): {desirable_men} patients.')
print(f'Normal HDL Range for women(50-59 mg/dL): {desirable_women} patients.
print(f'High HDL Range(60 mg/dL and above): {high} patients.')
```

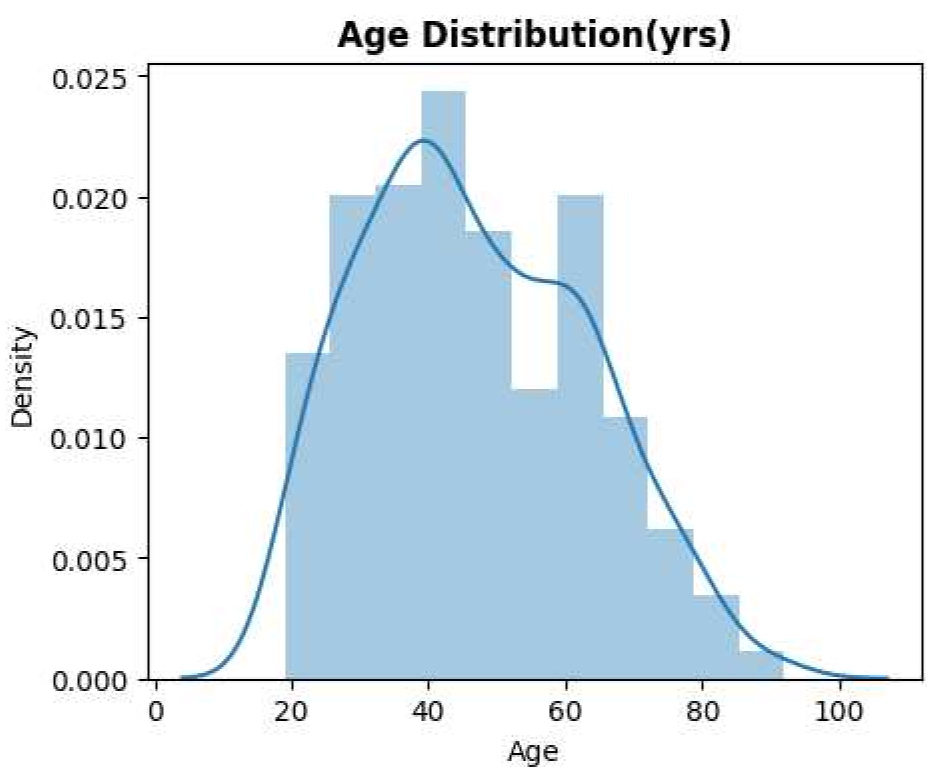
```
Low Range for men (below 40 mg/dL): 108 patients.
Low Range for women (below 50 mg/dL): 226 patients.
Normal HDL Range for men (40-59 mg/dL): 189 patients.
Normal HDL Range for women(50-59 mg/dL): 71 patients.
High HDL Range(60 mg/dL and above): 93 patients.
```



Age

In [13]:
```
plt.figure(figsize=(5,4))
sns.distplot(df.Age)
plt.title('Age Distribution(yrs)',fontdict={'fontweight': 'bold'})
```

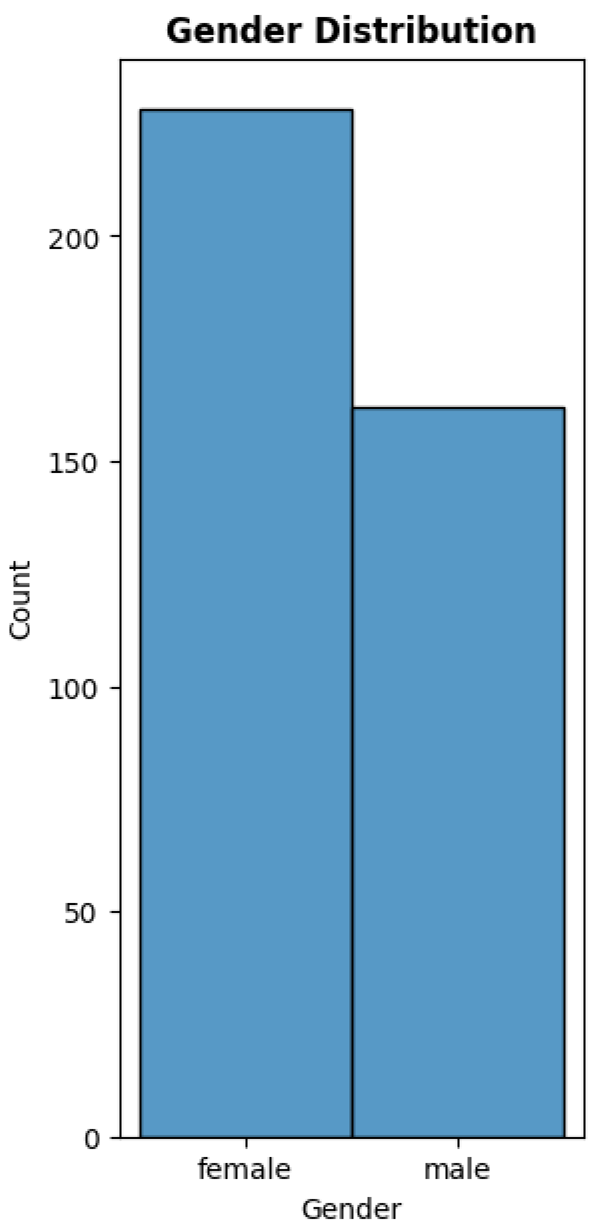
Out[13]: `Text(0.5, 1.0, 'Age Distribution(yrs)')`



Gender

In [14]:
```
plt.figure(figsize=(3,7))
sns.histplot(df.Gender)
plt.title('Gender Distribution',fontdict={'fontweight': 'bold'})
```

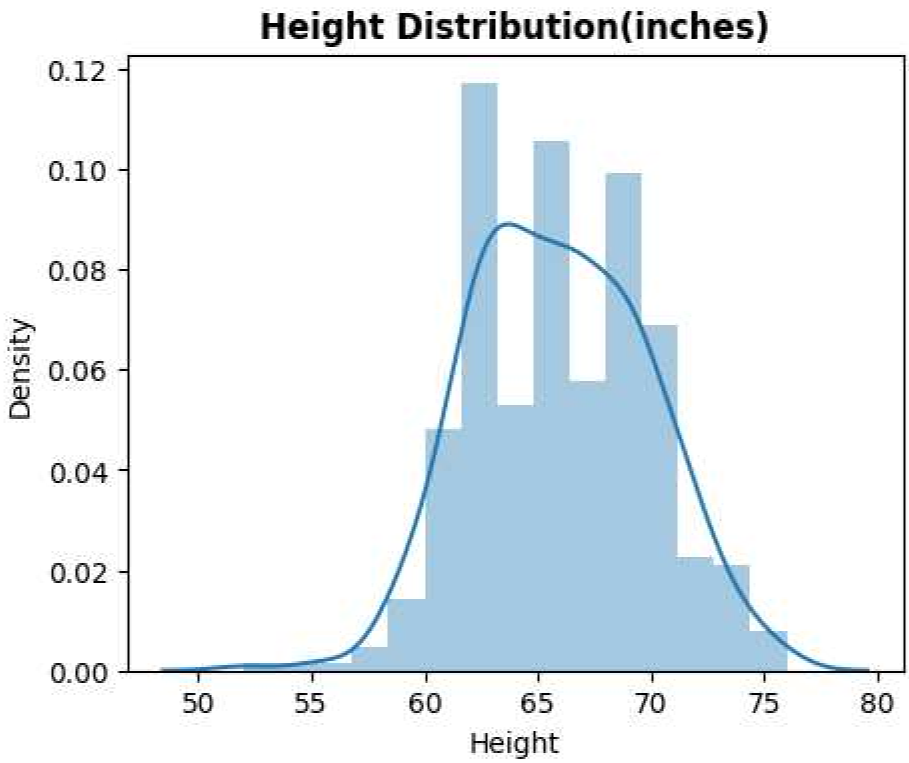
Out[14]: `Text(0.5, 1.0, 'Gender Distribution')`



Height

In [15]:
```
plt.figure(figsize=(5,4))
sns.distplot(df.Height)
plt.title('Height Distribution(inches)',fontdict={'fontweight': 'bold'})
```

Out[15]: `Text(0.5, 1.0, 'Height Distribution(inches)')`

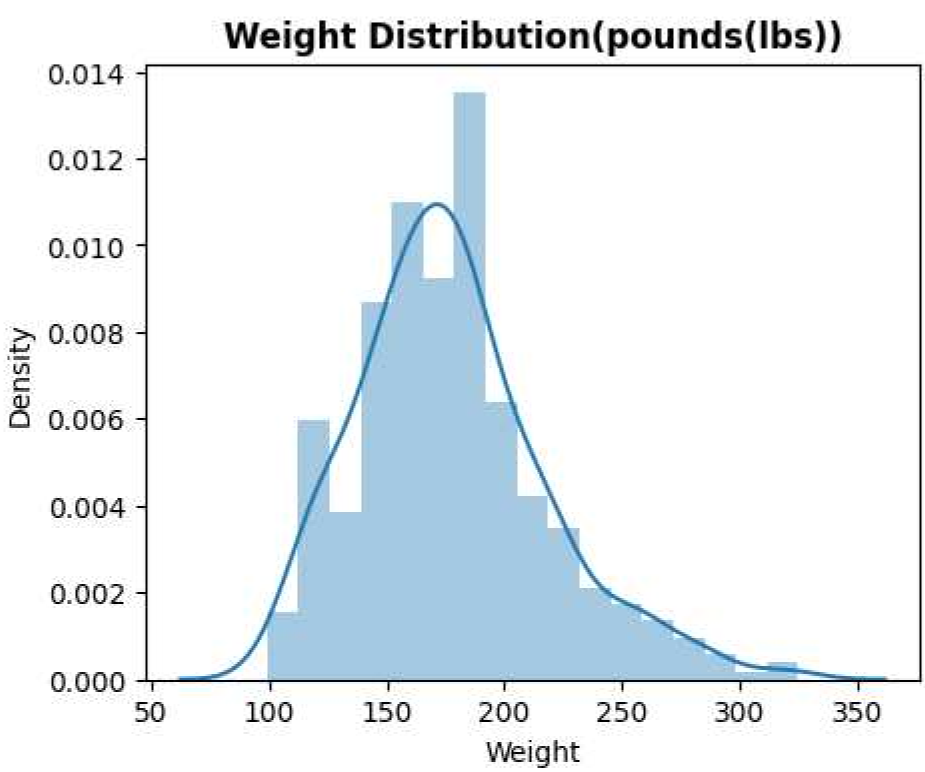


Weight

In [16]:

```
plt.figure(figsize=(5,4))
sns.distplot(df.Weight)
plt.title('Weight Distribution(pounds(lbs))',fontdict={'fontweight': 'bold'}
```

Out[16]: `Text(0.5, 1.0, 'Weight Distribution(pounds(lbs))')`



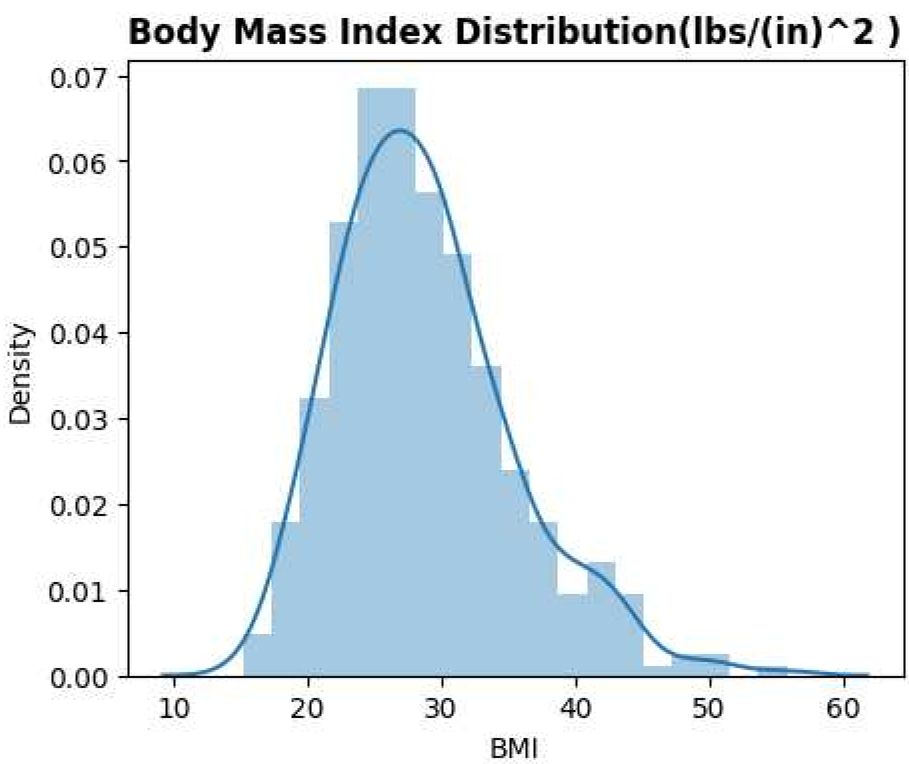
Body Mass Index

In [17]:
```
plt.figure(figsize=(5,4))
sns.distplot(df.BMI)
plt.title('Body Mass Index Distribution(lbs/(in)^2 )',fontdict={'fontweight'

underweight = df[df['BMI'] <18.5].count()[0]
normal_weight = df[(df['BMI'] >= 18.5) & (df['BMI'] < 24.9)].count()[0]
overweight = df[(df['BMI'] >=25.5)  & (df['BMI'] <30 )].count()[0]
obese = df[df['BMI'] >=30 ].count()[0]

print(f'Underweight (below 18.5 lbs/in^2): {underweight} patients.')
print(f'Normal weight (18.5-24.9 lbs/in^2): {normal_weight} patients.')
print(f'Overweight (25-29.9 lbs/in^2): {overweight} patients.')
print(f'Obese (30 lbs/in^2 and above): {obese} patients.')
```

```
Underweight (below 18.5 lbs/in^2): 9 patients.
Normal weight (18.5-24.9 lbs/in^2): 106 patients.
Overweight (25-29.9 lbs/in^2): 107 patients.
Obese (30 lbs/in^2 and above): 150 patients.
```



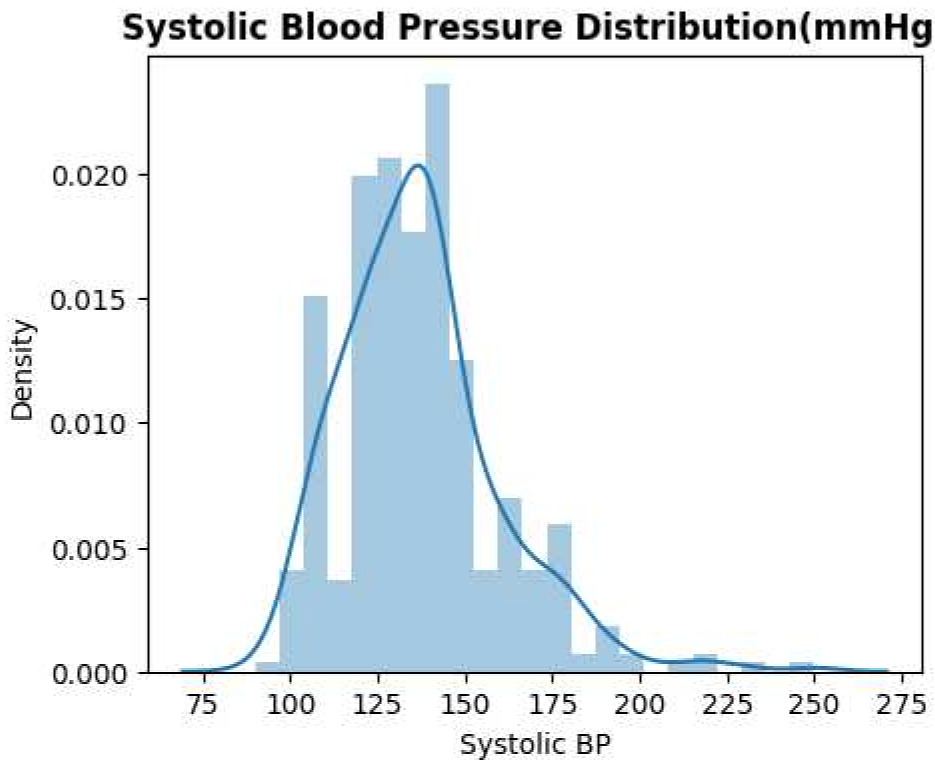
The BMI classes are as follows: underweight (BMI less than 18.5), normal weight (BMI between 18.5 and 24.9), overweight (BMI between 25 and 29.9), obesity (Class I, BMI between 30 and 34.9), obesity (Class II, BMI between 35 and 39.9), and obesity (Class III, BMI 40 or higher)

Systolic Blood Pressure

In [18]:
```
plt.figure(figsize=(5,4))
sns.distplot(df['Systolic BP'])
```

```
plt.title('Systolic Blood Pressure Distribution(mmHg)',fontdict={'fontweight
```

Out[18]: `Text(0.5, 1.0, 'Systolic Blood Pressure Distribution(mmHg)')`



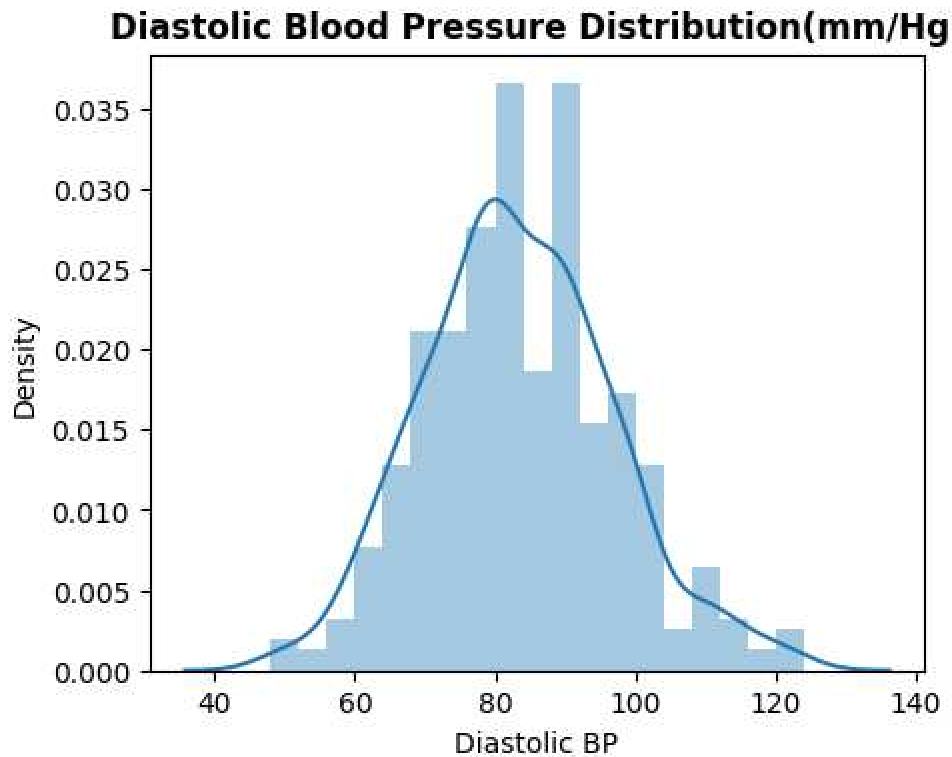
The systolic blood pressure (SBP) categories are as follows: normal (SBP below 120 mmHg but not up to 90mmHg), elevated (SBP between 120 and 129 mmHg), hypertension stage 1 (SBP between 130 and 139 mmHg), hypertension stage 2 (SBP 140 mmHg or higher), and hypertensive crisis (SBP higher than 180 mmHg and/or diastolic blood pressure (DBP) higher than 120 mmHg). This data has few portion below 120mmHg and the rest above.

Diastolic Blood Pressure

In [19]:

```
plt.figure(figsize=(5,4))
sns.distplot(df['Diastolic BP'])
plt.title('Diastolic Blood Pressure Distribution(mm/Hg)',fontdict={'fontweig
```

Out[19]: `Text(0.5, 1.0, 'Diastolic Blood Pressure Distribution(mm/Hg)')`

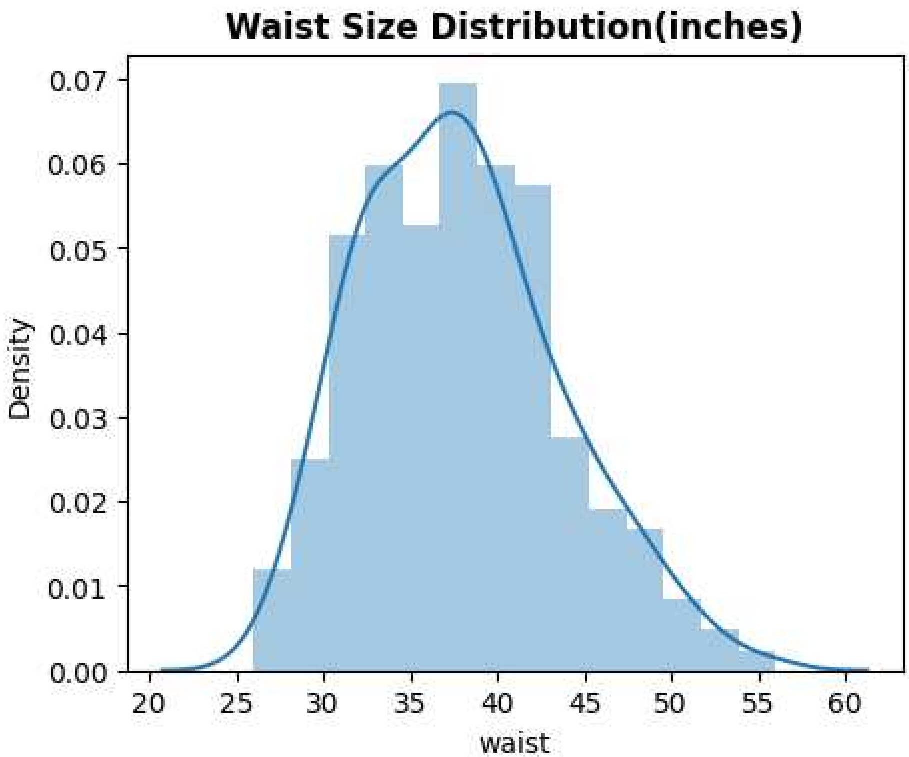


The diastolic blood pressure (DBP) categories are as follows: normal (DBP below 80 mmHg b ut not up to 60mmHg), elevated (DBP between 80 and 89 mmHg), hypertension stage 1 (DBP between 90 and 99 mmHg), hypertension stage 2 (DBP 100 mmHg or higher), and hypertensive crisis (DBP higher than 120 mmHg and/or systolic blood pressure (SBP) higher than 180 mmHg).

Waist size

In [20]:
```
plt.figure(figsize=(5,4))
sns.distplot(df['waist'])
plt.title('Waist Size Distribution(inches)',fontdict={'fontweight': 'bold'})
```

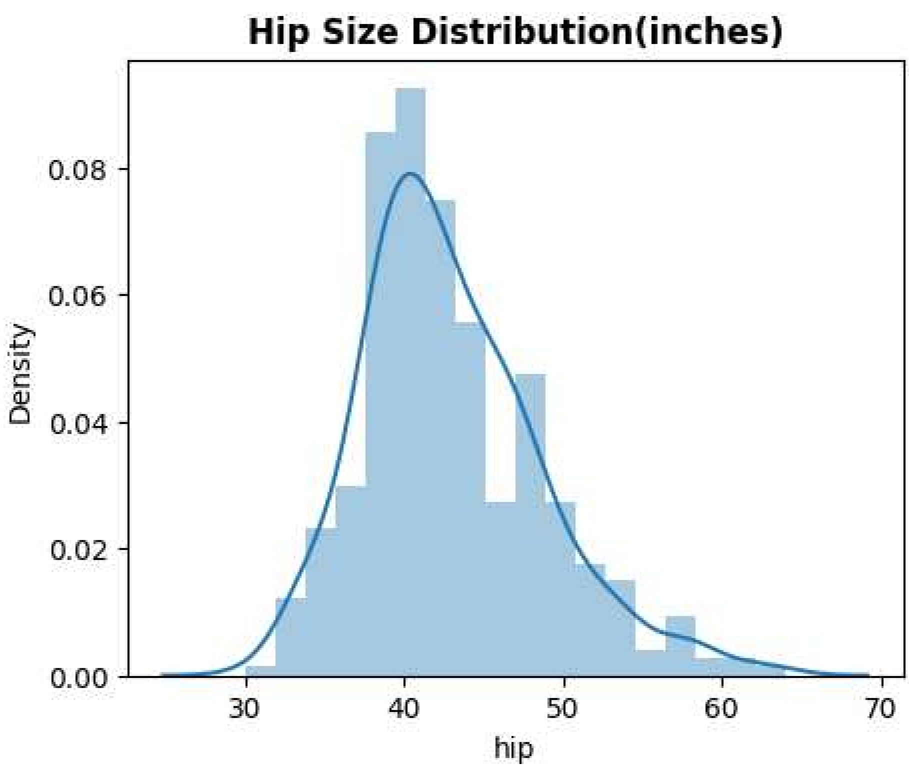
Out[20]:
```
Text(0.5, 1.0, 'Waist Size Distribution(inches)')
```



Hip Size

In [21]:
```
plt.figure(figsize=(5,4))
sns.distplot(df['hip'])
plt.title('Hip Size Distribution(inches)',fontdict={'fontweight': 'bold'})
```

Out[21]: `Text(0.5, 1.0, 'Hip Size Distribution(inches)')`



4 Correlation Analysis

4.1 Categorical Feature Analysis

In [22]:
```
from sklearn.preprocessing import LabelEncoder


label_encoder = LabelEncoder()
df['Diabetes'] = label_encoder.fit_transform(df['Diabetes'])
df['Gender'] = label_encoder.fit_transform(df['Gender'])
label_mapping = {0: 1, 1: 0}
df['Diabetes']=df['Diabetes'].map(label_mapping)

df.head()
```

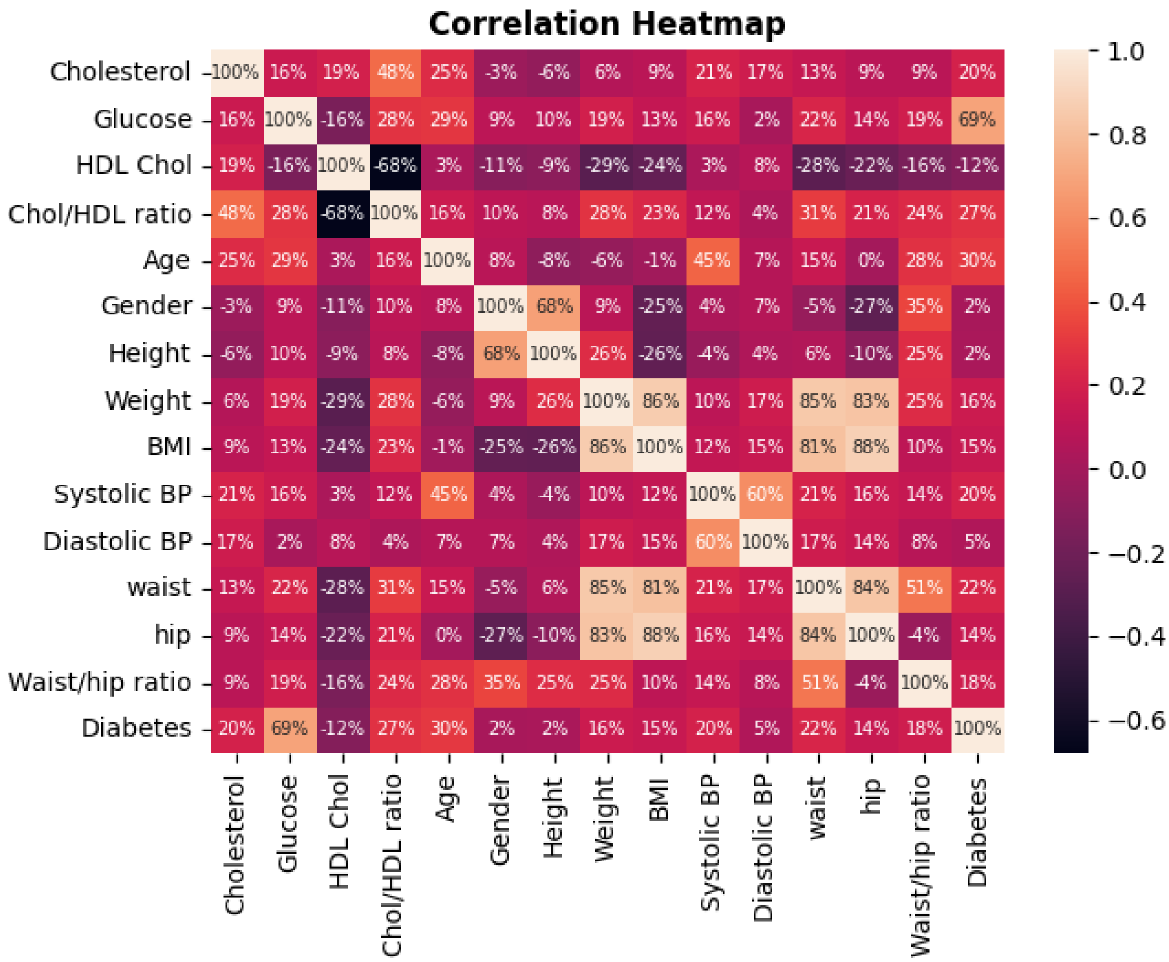
Out[22]:

Patient number	Cholesterol	Glucose	HDL Chol	Chol/HDL ratio	Age	Gender	Height	Weight	BMI	Systolic BP	Diastol B
1	193	77	49	3.9	19	0	61	119	22.5	118	7
2	146	79	41	3.6	19	0	60	135	26.4	108	5
3	217	75	54	4.0	20	0	67	187	29.3	110	7
4	226	97	70	3.2	20	0	64	114	19.6	122	6
5	164	91	67	2.4	20	0	70	141	20.2	122	8

4.2 Visualizing the correlation among the various features of the data

In [23]:
```
data_correlation= df.corr()
plt.figure(figsize= (7,5))
plt.title('Correlation Heatmap', fontdict= {'fontweight':'bold'})
sns.heatmap(data_correlation,annot=True,fmt='.0%' , annot_kws= {'fontsize':
```

Out[23]: `<AxesSubplot: title={'center': 'Correlation Heatmap'}>`

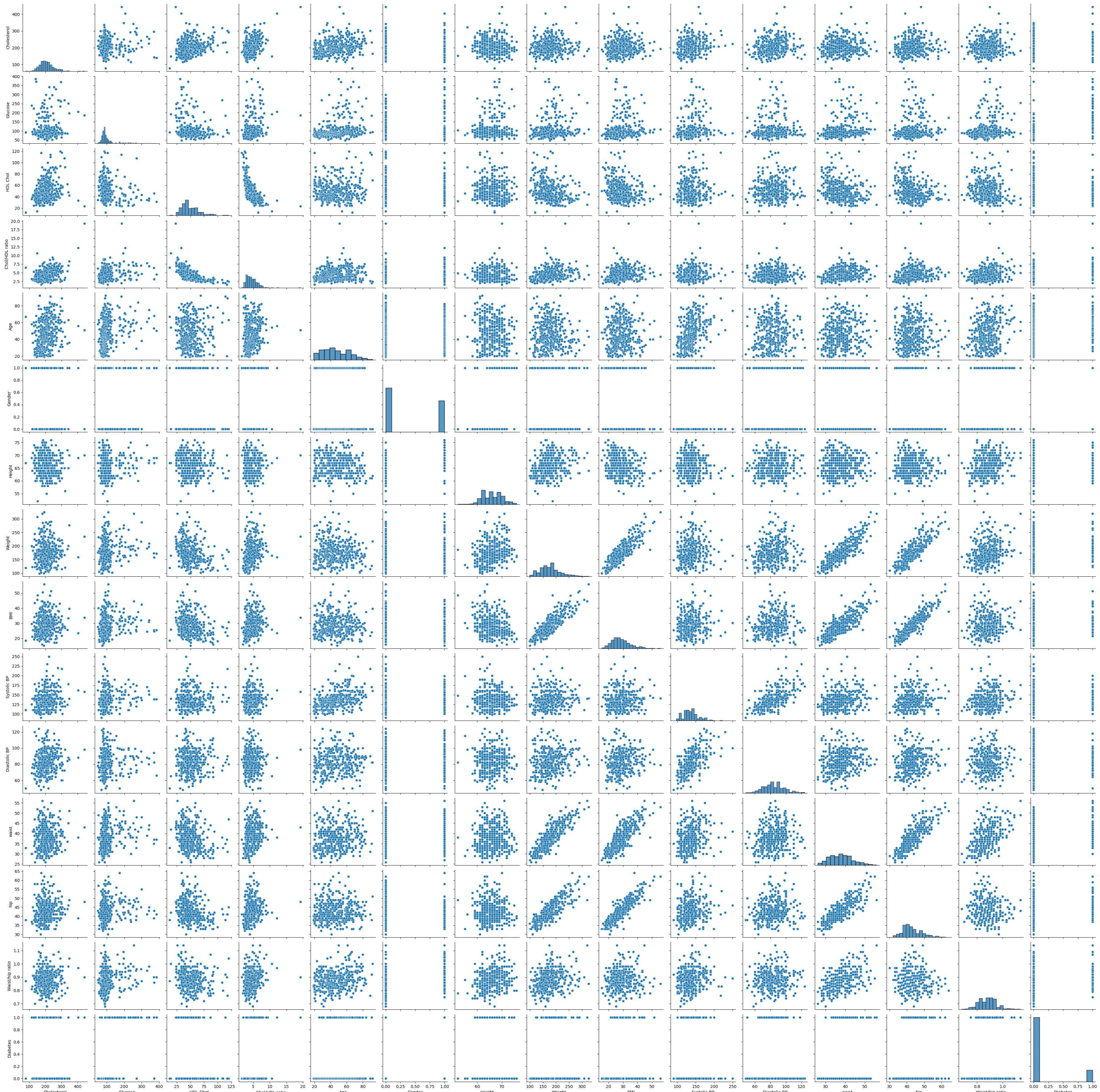


In [24]:

```
#cat= df['Diabetes_encoded','Gender_encoded']
#data= df.drop(cat)
plt.figure(figsize= (15,15))
sns.pairplot(df)
```

Out[24]:

```
<seaborn.axisgrid.PairGrid at 0x282415ddd0>
<Figure size 1500x1500 with 0 Axes>
```



Let us take a look at features with very good correlation.

In [25]:
```
correlation_matrix = df.corr()

high_correlation_pairs = []
for i in range(len(correlation_matrix.columns)):
    for j in range(i + 1, len(correlation_matrix.columns)):
        if abs(correlation_matrix.iloc[i, j]) > 0.30:
            pair = (correlation_matrix.columns[i], correlation_matrix.column
            high_correlation_pairs.append(pair)


print(high_correlation_pairs)
```

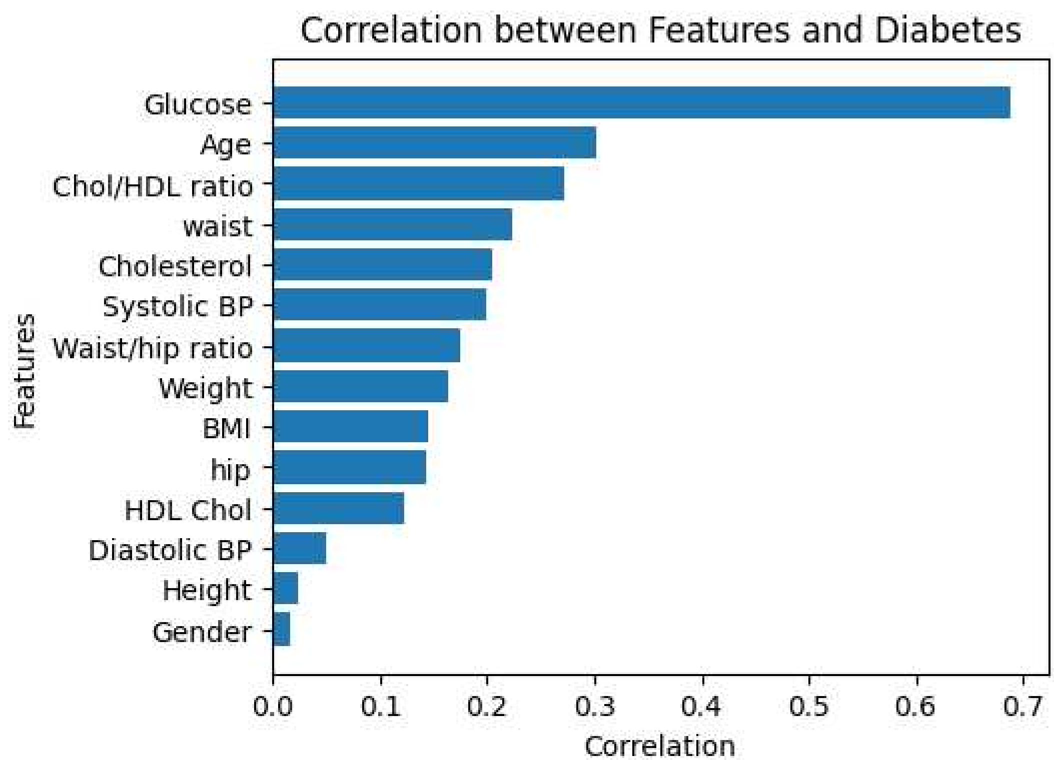
```
[('Cholesterol', 'Chol/HDL ratio'), ('Glucose', 'Diabetes'), ('HDL Chol', 'C
hol/HDL ratio'), ('Chol/HDL ratio', 'waist'), ('Age', 'Systolic BP'), ('Ag
e', 'Diabetes'), ('Gender', 'Height'), ('Gender', 'Waist/hip ratio'), ('Weig
ht', 'BMI'), ('Weight', 'waist'), ('Weight', 'hip'), ('BMI', 'waist'), ('BM
I', 'hip'), ('Systolic BP', 'Diastolic BP'), ('waist', 'hip'), ('waist', 'Wa
ist/hip ratio')]
```

The pairs above show features that have correlation coeffient above plus or minus 0.3 .

4.3 Visualizing correlation between features and target(Diabetes column)

In [26]:

```
correlation_matrix = df.corr()
target_correlation = correlation_matrix['Diabetes']
target_correlation = target_correlation.abs().sort_values(ascending=True)
target_correlation = target_correlation.drop('Diabetes')
plt.figure(figsize=(5, 4))
plt.barh(target_correlation.index, target_correlation.values)
plt.xlabel('Correlation')
plt.ylabel('Features')
plt.title('Correlation between Features and Diabetes')
plt.show()
```



From the graph we can see that glucose level has the highest correlation coeffeicient with Diabetes, then surprisingly, age through to height having the lowest correlation coefficient with Diabetes.

4. 4 Exploring new features

Since the age seems to have a good correlation, let us utilise it in exploring new features with higher correlation.

In [27]:

```
#creating square of the ages feature
age_square = (df.Age)**2
New_features = pd.DataFrame()
New_features['Age_square']= age_square
```

Waist seems to have a good correlation with Diabetes, I am creating a new feature ((waist)^2/Age) to see how it will correlate with diabetes since a relationship between a person's waist and age can reveals obesity.

In [28]:
```
Square_Waist_per_Age= df['waist']**2/df['Age']
New_features['Square_Waist/Age']= Square_Waist_per_Age
```

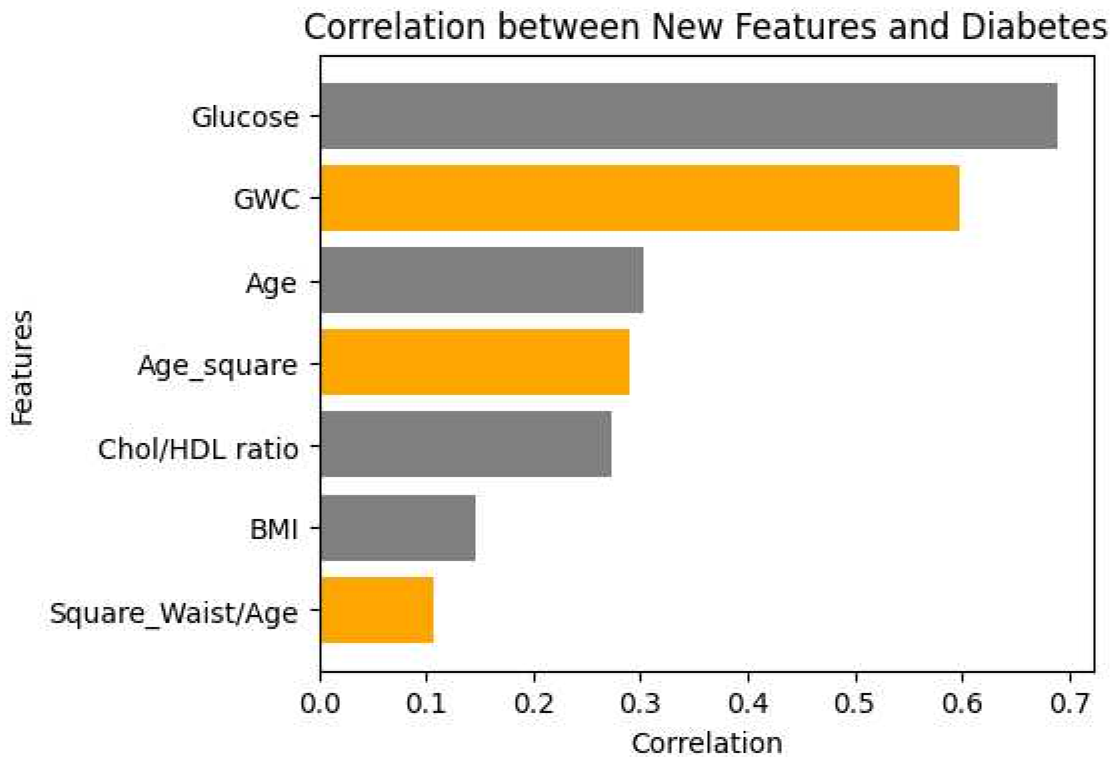
Creating another feature which is square root of (Glucose level * Weight * (Chol/HDL))

In [29]:
```
GWC= np.sqrt(df.Glucose * df.Weight * df['Chol/HDL ratio'])
New_features['GWC']= GWC
```

In [30]:
```
New_features['Diabetes']= df.Diabetes
New_features['Glucose']=df.Glucose
New_features['BMI']=df.BMI
New_features['Chol/HDL ratio'] =df['Chol/HDL ratio']
New_features['Age']= df.Age

corr_matrix = New_features.corr()
target_correlation = corr_matrix['Diabetes']
target_correlation = target_correlation.abs().sort_values(ascending=True)
target_correlation = target_correlation.drop('Diabetes')
plt.figure(figsize=(5, 4))
color=['orange','grey','grey','orange','grey','orange','grey']
plt.barh(target_correlation.index, target_correlation.values, color=color)
plt.xlabel('Correlation')
plt.ylabel('Features')
plt.title('Correlation between New Features and Diabetes')
plt.show()

print(f'Correlation Coefficient of GWC is  {target_correlation.GWC} whilst t
```




```
Correlation Coefficient of GWC is  0.5975911038445685 whilst that of glucose
is 0.6890795038664445
```

From the graph, it could be revealed that the new feature GWC which is the square root of (Weight * Glucose Level * (cholesterol level/HDL level) tends to be of a very high correlation coefficient with Diabetes and can be a very useful factor in diagnosing diabetes even over most of the the traditional factors that have always been used. The feature GWC makes use of glucose level, cholesterol level, weight and HDL level to give a more accurate foresight concerning the likelihood of the patient being diabetic or not. From analysis, GWC above 200 calls for attention since it is more likely patient is diabetic.

5 Prediction Model

In [31]:

```
data= New_features.merge(df)
X = data.drop(columns ='Diabetes')
y = data.Diabetes

X_train,X_test,y_train,y_valid= train_test_split(X,y,test_size=0.2,random_st

my_model= RandomForestClassifier(n_estimators=500 , max_depth= 100, random_s
my_model.fit(X_train,y_train)
predictions = my_model.predict(X_test)

efficiency = my_model.score(X_test, y_valid)
mae= mean_absolute_error(y_valid,predictions)
print('model accuracy:', efficiency)
print('mean absolute error:', mae)
```

```
model accuracy: 0.9230769230769231
mean absolute error: 0.07692307692307693
```

5.2Testing the model

```
In [32]: feature= np.array([[2400,29,350,83,24,(200/39),45,200,39,0,52,170,122,75,33,
my_model.predict(feature)
```

```
Out[32]: array([0], dtype=int64)
```

5. 3 Creating an automated diagnosing predictor

```
In [36]: import pandas as pd
import numpy as np

def get_numeric_input(prompt):
    while True:
        try:
            value = int(input(prompt))
            return value
        except ValueError:
            print("Invalid input. Please enter a numeric value.")

def predict_diabetes():
    age = get_numeric_input('Age of patient: ')
    age_sqr = age ** 2
    waist = get_numeric_input('Waist size in Inches: ')
    weight = get_numeric_input('Weight of patient in lbs: ')
    sqr_waist_per_age = (waist ** 2) / age
    glucose = get_numeric_input('Fasting glucose level in mg/dL: ')
    cholesterol = get_numeric_input('Cholesterol level in mg/dL: ')
    HDL = get_numeric_input('HDL Cholesterol level in mg/dL: ')
    GWC = np.sqrt(glucose * weight * (cholesterol / HDL))
    while True:
        try:
            gender = int(input("Gender (Input '1' for male or '0' for female
            gender = str(gender)
            if gender not in ('0', '1'):
                raise ValueError
            break
        except ValueError:
            print("Invalid input. Please enter either '0' or '1'.")
    height = get_numeric_input('Height of patient in inches: ')
    BMI = 703 * (weight / (height ** 2))
    systolic_bp = get_numeric_input('Systolic Blood Pressure: ')
    diastolic_bp = get_numeric_input('Diastolic Blood Pressure: ')
    hip = get_numeric_input('Hip size in inches: ')
    waist_per_hip = waist / hip
    cholesterol_per_HDL = cholesterol / HDL

    # Forming a dataframe for the model
    data = {
        'Age_square': [age_sqr],
        'Square_Waist/Age': [sqr_waist_per_age],
```

```python
        'GWC': [GWC],
        'Glucose': [glucose],
        'BMI': [BMI],
        'Chol/HDL ratio': [cholesterol_per_HDL],
        'Age': [age],
        'Cholesterol': [cholesterol],
        'HDL Chol': [HDL],
        'Gender': [gender],
        'Height': [height],
        'Weight': [weight],
        'Systolic BP': [systolic_bp],
        'Diastolic BP': [diastolic_bp],
        'waist': [waist],
        'hip': [hip],
        'Waist/hip ratio': [waist_per_hip]
    }

    df = pd.DataFrame(data)

    prediction = my_model.predict(df)

    if prediction == 1:
        return "This patient is likely to be diabetic."
    else:
        return "This patient is likely to be undiabetic."



result = predict_diabetes()
print(result)
```

```
Age of patient: 43
Waist size in Inches: 41
Weight of patient in lbs: 200
Fasting glucose level in mg/dL: 127
Cholesterol level in mg/dL: 293
HDL Cholesterol level in mg/dL: 60
Gender (Input '1' for male or '0' for female): 1
Height of patient in inches: 65
Systolic Blood Pressure: 141
Diastolic Blood Pressure: 97
Hip size in inches: 39
This patient is likely to be undiabetic.
```

In [ ]:

```python
result = predict_diabetes()
print(result)
```

In [ ]: