To study the smoking effects on metabolic health and Inflammation Markers

Objectives:

To build a model which predicts diabetic category using smokers and non smokers body signal

Comparing the features of smokers and non smokers usind exploratory data analysis.

To study the associstion of smoking on blood sugar level and dental caries.

Determing whether smokers cholestrol level influencing on blood pressure.

In [1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

ID : serial number given to a person.

gender: the gender of a person being either female(0) or male(1).

age : 5-years gap age groups.

height(cm)

weight(kg)

waist(cm) : Waist circumference length

eyesight(left): Denoted by a value between 0.1-2.5, or 9.9

eyesight(right): Denoted by a value between 0.1-2.5, or 9.9

hearing(left): hearing of the person's ear

hearing(right): hearing of the person's ear

systolic : blood pressure.

relaxation : blood pressure.

fasting blood sugar: blood sugar before meals level

Cholesterol : total

triglyceride

HDL : cholesterol type

LDL : cholesterol type

hemoglobin

serum creatinine

AST : glutamic oxaloacetic transaminase type

ALT : glutamic oxaloacetic transaminase type

Gtp : γ-GTP

oral : Oral Examination status (s whether the examinee accepted the oral examination).

dental caries

tartar : tartar status

smoking: smoking status of a person

In [2]:
```
df=pd.read_csv("smoking.csv")
df
```

Out[2]:

	ID	smoking	gender	age	height(cm)	weight(kg)	waist(cm)	eyesight(left)	eyesigh
0	0	No	F	40	155	60	81.3	1.2	
1	1	No	F	40	160	60	81.0	0.8	
2	2	Yes	M	55	170	60	80.0	0.8	
3	3	No	M	40	165	70	88.0	1.5	
4	4	No	F	40	155	60	86.0	1.0	
...	...	...	...	...	...	...	...	...	
55687	55676	No	F	40	170	65	75.0	0.9	
55688	55681	No	F	45	160	50	70.0	1.2	
55689	55683	No	F	55	160	50	68.5	1.0	
55690	55684	No	M	60	165	60	78.0	0.8	
55691	55691	Yes	M	55	160	65	85.0	0.9	

55692 rows × 26 columns

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 55692 entries, 0 to 55691
Data columns (total 26 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   ID                   55692 non-null  int64  
 1   smoking              55692 non-null  object 
 2   gender               55692 non-null  object 
 3   age                  55692 non-null  int64  
 4   height(cm)           55692 non-null  int64  
 5   weight(kg)           55692 non-null  int64  
 6   waist(cm)            55692 non-null  float64
 7   eyesight(left)       55692 non-null  float64
 8   eyesight(right)      55692 non-null  float64
 9   hearing(left)        55692 non-null  object 
 10  hearing(right)       55692 non-null  object 
 11  systolic             55692 non-null  int64  
 12  relaxation           55692 non-null  int64  
 13  fasting blood sugar  55692 non-null  int64  
 14  Cholesterol          55692 non-null  int64  
 15  triglyceride         55692 non-null  int64  
 16  HDL                  55692 non-null  int64  
 17  LDL                  55692 non-null  int64  
 18  hemoglobin           55692 non-null  float64
 19  serum creatinine     55692 non-null  float64
 20  AST                  55692 non-null  int64  
 21  ALT                  55692 non-null  int64  
 22  Gtp                  55692 non-null  int64  
 23  oral                 55692 non-null  object 
 24  dental caries        55692 non-null  int64  
 25  tartar               55692 non-null  object 
dtypes: float64(5), int64(15), object(6)
memory usage: 11.0+ MB
```

In [4]:
```
df.isnull().sum()
```

Out[4]:
```
ID                     0
smoking                0
gender                 0
age                    0
height(cm)             0
weight(kg)             0
waist(cm)              0
eyesight(left)         0
eyesight(right)        0
hearing(left)          0
hearing(right)         0
systolic               0
relaxation             0
fasting blood sugar    0
Cholesterol            0
triglyceride           0
HDL                    0
LDL                    0
hemoglobin             0
serum creatinine       0
AST                    0
ALT                    0
Gtp                    0
oral                   0
dental caries          0
tartar                 0
dtype: int64
```

In [5]:
```
df.columns
```

Out[5]:
```
Index(['ID', 'smoking', 'gender', 'age', 'height(cm)', 'weight(kg)',
       'waist(cm)', 'eyesight(left)', 'eyesight(right)', 'hearing(left)',
       'hearing(right)', 'systolic', 'relaxation', 'fasting blood sugar',
       'Cholesterol', 'triglyceride', 'HDL', 'LDL', 'hemoglobin',
       'serum creatinine', 'AST', 'ALT', 'Gtp', 'oral', 'dental caries',
       'tartar'],
      dtype='object')
```

In [6]:
```
def sys_cat(sys_bp):
    if sys_bp>=0 and sys_bp<=90:
        return "Hypotension"
    elif sys_bp>=90 and sys_bp<=120:
        return "Normal"
    elif sys_bp>=120 and sys_bp<=130:
        return "Elevated"
    elif sys_bp>=130 and sys_bp<=140:
        return "Hypertension stage1"
    elif sys_bp>=140 and sys_bp<=150:
        return "Hypertension stage2"
    else:
        return "Hypertension crisis"

df['systolic_category']=df['systolic'].apply(sys_cat)
```

In [7]:
```
def rel_cat(rel_bp):
    if rel_bp>=0 and rel_bp<=60:
        return "Hypotension"
    elif rel_bp>=60 and rel_bp<=80:
        return "Normal"
    elif rel_bp>=80 and rel_bp<=90:
        return "Elevated"
    elif rel_bp>=90 and rel_bp<=100:
        return "Hypertension stage1"
    elif rel_bp>=100 and rel_bp<=120:
        return "Hypertension stage2"
    else:
        return "Hypertension crisis"

df['relaxation_category']=df['relaxation'].apply(rel_cat)
```

In [8]:
```
def diab_cat(sugar):
    if sugar>=0 and sugar<=80:
        return "Hypoglycemia"
    elif sugar>=80 and sugar<=100:
        return "Normal"
    elif sugar>=100 and sugar<=130:
        return "Pre-Diabetic"
    else:
        return "Diabetic"

df['diabetic_category']=df['fasting blood sugar'].apply(diab_cat)
```

In [9]:
```
df['dental caries'] = df['dental caries'].replace({0: 'Yes', 1: 'No'})

```

In [10]:
```
df=df.loc[:,['ID', 'smoking', 'gender', 'age', 'height(cm)', 'weight(kg
         'waist(cm)', 'eyesight(left)', 'eyesight(right)', 'hearing(left)
         'hearing(right)', 'systolic','systolic_category', 'relaxation','
         'diabetic_category','Cholesterol', 'triglyceride', 'HDL', 'LDL',
         'serum creatinine', 'AST', 'ALT', 'Gtp', 'oral', 'dental caries'
         'tartar']]
df
```

Out[10]:

	ID	smoking	gender	age	height(cm)	weight(kg)	waist(cm)	eyesight(left)	eyesigh
0	0	No	F	40	155	60	81.3	1.2	
1	1	No	F	40	160	60	81.0	0.8	
2	2	Yes	M	55	170	60	80.0	0.8	
3	3	No	M	40	165	70	88.0	1.5	
4	4	No	F	40	155	60	86.0	1.0	
...	...	...	...	...	...	...	...	...	
55687	55676	No	F	40	170	65	75.0	0.9	
55688	55681	No	F	45	160	50	70.0	1.2	
55689	55683	No	F	55	160	50	68.5	1.0	
55690	55684	No	M	60	165	60	78.0	0.8	
55691	55691	Yes	M	55	160	65	85.0	0.9	

55692 rows × 29 columns

In [ ]:
```

```

In [ ]:
```

```

Exploratary Data Analysis

In [11]:
```
df["smoking"].value_counts()
```

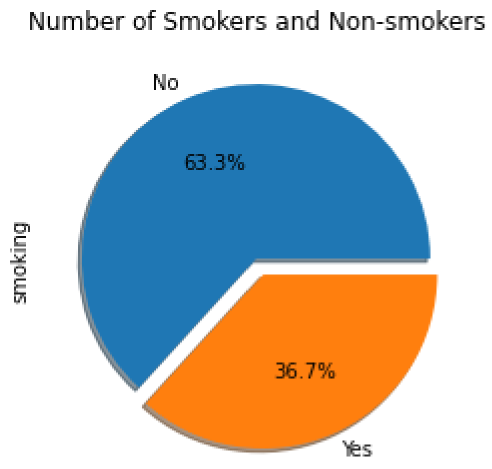
Out[11]:
```
No     35237
Yes    20455
Name: smoking, dtype: int64
```

In [12]:
```
df["smoking"].value_counts()
```

Out[12]:
```
No     35237
Yes    20455
Name: smoking, dtype: int64
```

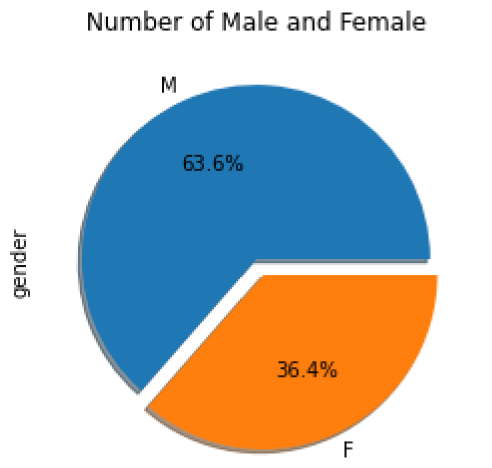
Total number of smokers and non-smokers

In [13]:
```
df["smoking"].value_counts().plot.pie(shadow=True,explode=[0,0.1],autop
plt.title("Number of Smokers and Non-smokers")
plt.show()
```



In [14]:
```
# only 36.7% people smoke
# 63.3% people do not smoke
```

In [15]:
```
df["gender"].value_counts().plot.pie(shadow=True,explode=[0,0.1],autopct
plt.title("Number of Male and Female")
plt.show()
```



In [16]:
```
# Total 63.6% are male
# 36.4% are female
```

In [17]:
```
smokeYes=df[df["smoking"]=="Yes"]
smokeYes.head(2)
```

Out[17]:

	ID	smoking	gender	age	height(cm)	weight(kg)	waist(cm)	eyesight(left)	eyesight(right)
2	2	Yes	M	55	170	60	80.0	0.8	0.8
6	6	Yes	M	40	160	60	85.5	1.0	1.0

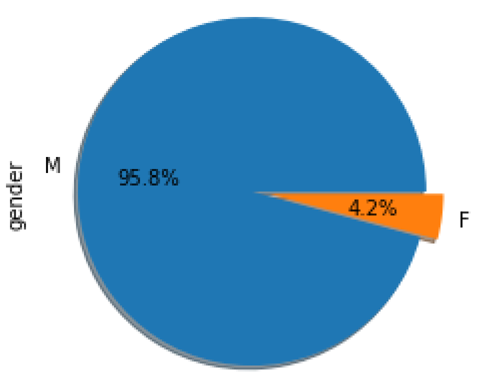
2 rows × 29 columns

In [18]:
```
smokeYes['gender'].value_counts()
```

Out[18]:
```
M    19596
F      859
Name: gender, dtype: int64
```

In [19]:
```
smokeYes["gender"].value_counts().plot.pie(shadow=True,explode=[0,0.1],
plt.title("Number of male and female smokers")
plt.show()

```

Number of male and female smokers



Average age of male and female smokers.

In [20]:
```
smokeNo=df[df["smoking"]=="No"]
#smokeNo
```

In [21]:
```
smokeYesW=smokeYes[smokeYes["gender"]=="F"]
#smokeYesW
```

In [22]:
```
smokeYesM=smokeYes[smokeYes["gender"]=="M"]
smokeYesM
```

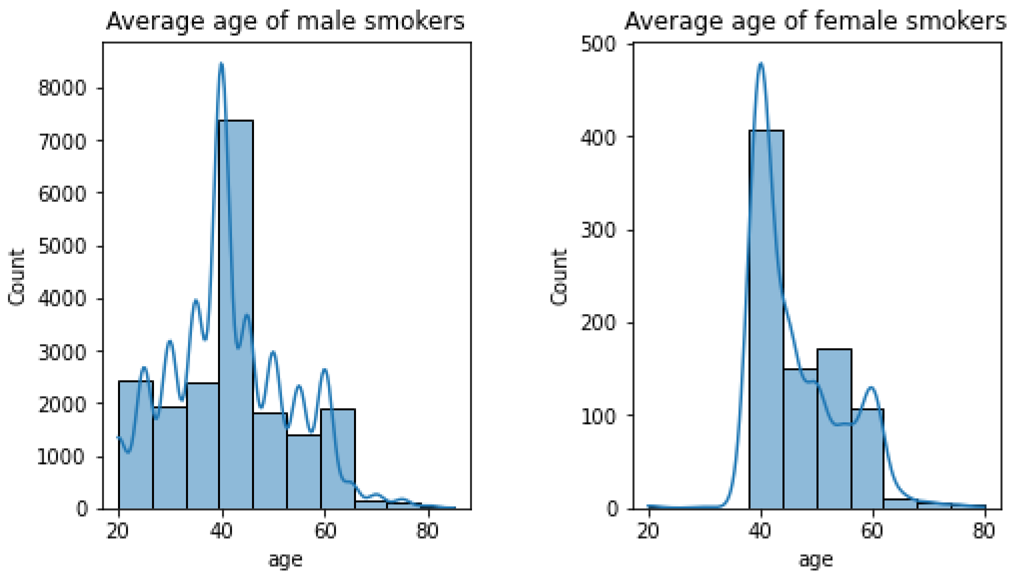
Out[22]:

	ID	smoking	gender	age	height(cm)	weight(kg)	waist(cm)	eyesight(left)	eyesigh
2	2	Yes	M	55	170	60	80.0	0.8	
6	6	Yes	M	40	160	60	85.5	1.0	
12	13	Yes	M	35	170	70	81.0	1.5	
17	19	Yes	M	35	165	70	87.5	1.0	
18	21	Yes	M	60	165	65	79.0	1.0	
...	...	...	...	...	...	...	...	...	
55673	55596	Yes	M	75	155	45	68.0	0.6	
55675	55607	Yes	M	30	170	70	88.8	1.0	
55679	55632	Yes	M	45	180	90	92.0	1.2	
55684	55666	Yes	M	40	170	65	85.0	1.2	
55691	55691	Yes	M	55	160	65	85.0	0.9	

19596 rows × 29 columns

In [23]:
```
plt.figure(figsize=(8,5))

plt.subplot(1,2,1)
sns.histplot(data=smokeYesM,x='age',kde=True,bins=10)
plt.title("Average age of male smokers")

plt.subplot(1,2,2)

sns.histplot(data=smokeYesW,x='age',kde=True,bins=10)
plt.title("Average age of female smokers")



plt.tight_layout(4)
plt.show()
```

```
C:\Users\darsh\AppData\Local\Temp/ipykernel_18136/3942839391.py:14: Matplot
libDeprecationWarning: Passing the pad parameter of tight_layout() position
ally is deprecated since Matplotlib 3.3; the parameter will become keyword-
only two minor releases later.
  plt.tight_layout(4)
```



In [24]:
```
w=sum(smokeYesW['age'])/len(smokeYesW['age'])
print('Average age of female smokers',w)
```

```
Average age of female smokers 46.38533178114086
```

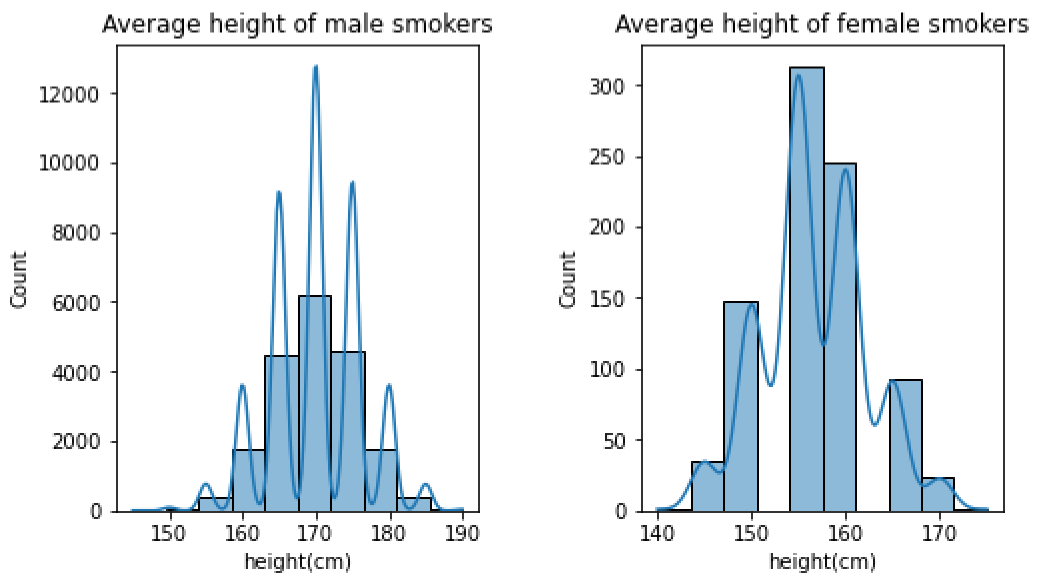
In [25]:
```
m=sum(smokeYesM['age'])/len(smokeYesM['age'])
print('Average age male smokers',m)
```

```
Average age male smokers 41.39798938558889
```

Average height of male and female smokers.

In [26]:
```python
plt.figure(figsize=(8,5))

plt.subplot(1,2,1)
sns.histplot(data=smokeYesM,x='height(cm)',kde=True,bins=10)
plt.title("Average height of male smokers")



plt.subplot(1,2,2)
sns.histplot(data=smokeYesW,x='height(cm)',kde=True,bins=10)
plt.title("Average height of female smokers")

plt.tight_layout(4)
plt.show()
```

```
C:\Users\darsh\AppData\Local\Temp/ipykernel_18136/1465316697.py:13: Matplot
libDeprecationWarning: Passing the pad parameter of tight_layout() position
ally is deprecated since Matplotlib 3.3; the parameter will become keyword-
only two minor releases later.
  plt.tight_layout(4)
```



In [27]:
```python
w=sum(smokeYesW['height(cm)'])/len(smokeYesW['height(cm)'])
print('Average height of female smokers',w)
```

```
Average height of female smokers 156.64726426076834
```

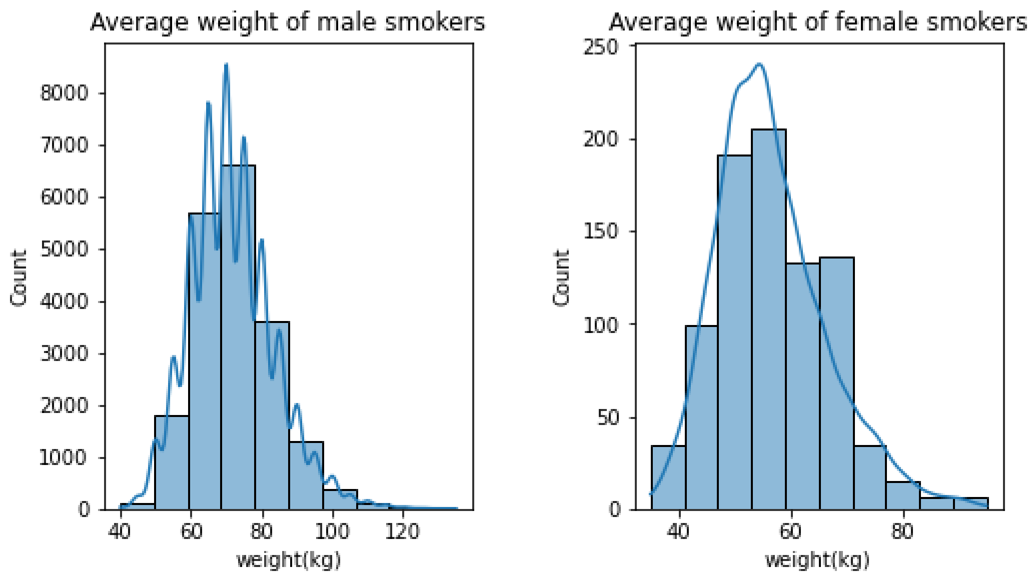
In [28]:
```python
m=sum(smokeYesM['height(cm)'])/len(smokeYesM['height(cm)'])
print('Average Height of male smokers',m)
```

```
Average Height of male smokers 169.996938150643
```

Average weight male and female smokers

In [29]:
```python
plt.figure(figsize=(8,5))

plt.subplot(1,2,1)
sns.histplot(data=smokeYesM,x='weight(kg)',kde=True,bins=10)
plt.title("Average weight of male smokers")

plt.subplot(1,2,2)
sns.histplot(data=smokeYesW,x='weight(kg)',kde=True,bins=10)
plt.title("Average weight of female smokers")



plt.tight_layout(4)
plt.show()
```

```
C:\Users\darsh\AppData\Local\Temp/ipykernel_18136/2753398324.py:13: Matplot
libDeprecationWarning: Passing the pad parameter of tight_layout() position
ally is deprecated since Matplotlib 3.3; the parameter will become keyword-
only two minor releases later.
  plt.tight_layout(4)
```



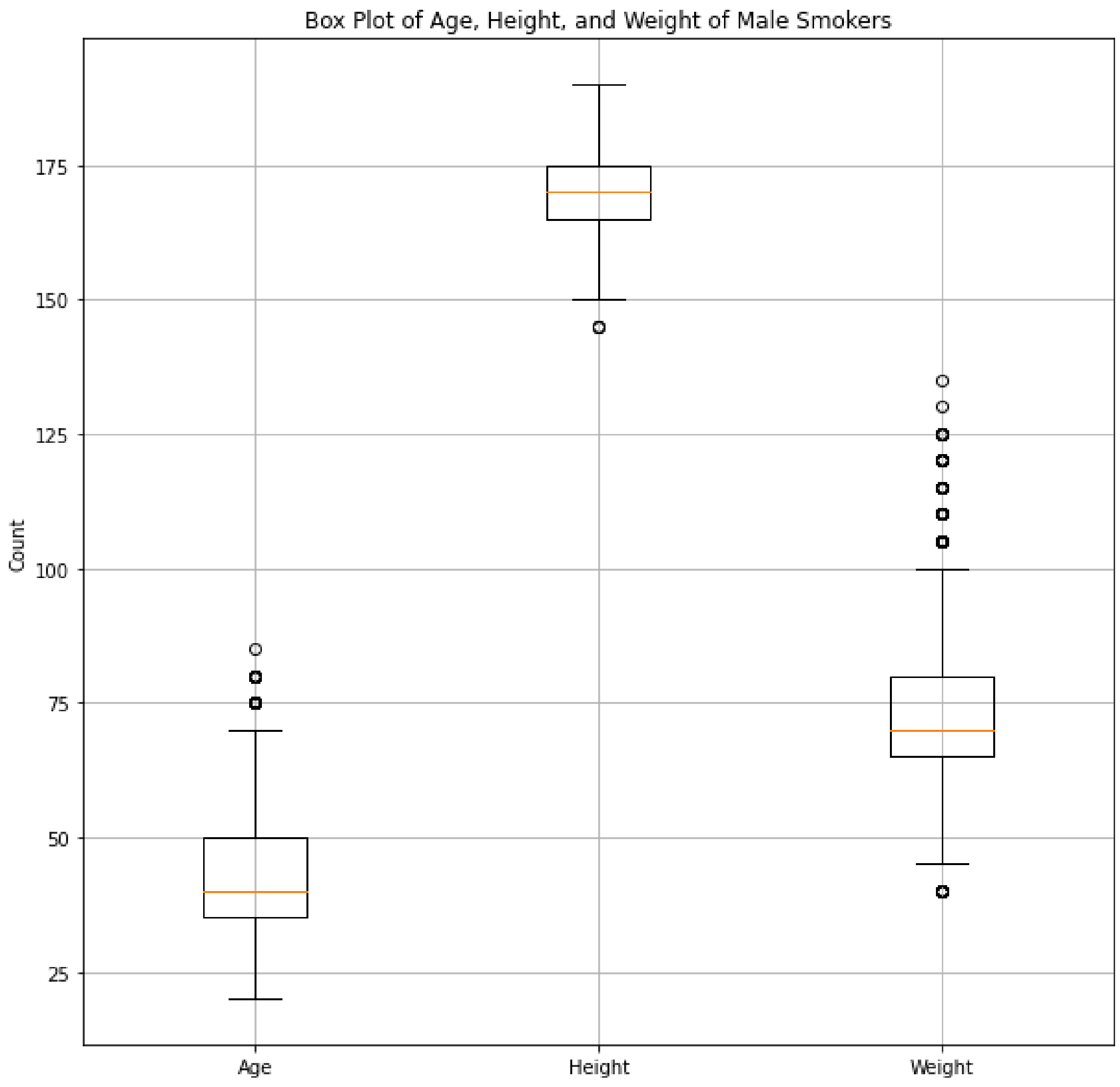
In [30]:
```python
w=sum(smokeYesW['weight(kg)'])/len(smokeYesW['weight(kg)'])
print('Average weight of female smokers',w)
```

```
Average weight of female smokers 56.44935972060536
```

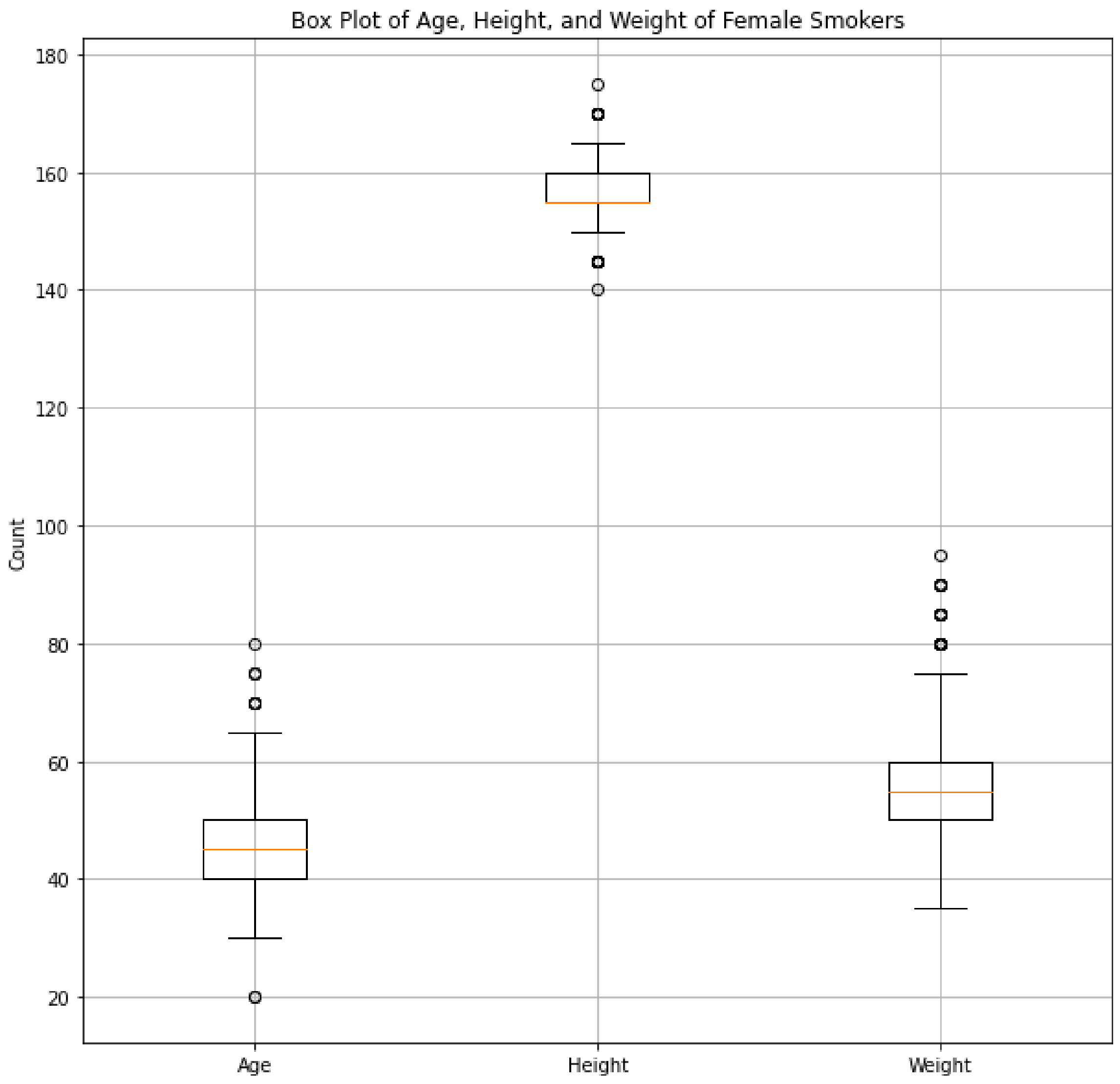
In [31]:
```python
m=sum(smokeYesM['weight(kg)'])/len(smokeYesM['weight(kg)'])
print('Average weight of female smokers',m)
```

```
Average weight of female smokers 71.59573382322922
```

In [32]:
```
plt.figure(figsize=(10,10))

plt.boxplot([smokeYesM['age'],smokeYesM['height(cm)'], smokeYesM['weigh
plt.title('Box Plot of Age, Height, and Weight of Male Smokers')
plt.ylabel('Count')
plt.grid(True)
plt.show()
```



In [33]:
```python
plt.figure(figsize=(10,10))

plt.boxplot([smokeYesW['age'],smokeYesW['height(cm)'], smokeYesW['weigh
plt.title('Box Plot of Age, Height, and Weight of Female Smokers')
plt.ylabel('Count')
plt.grid(True)
plt.show()
```



In [34]:
```python
smokeYes.head()
```

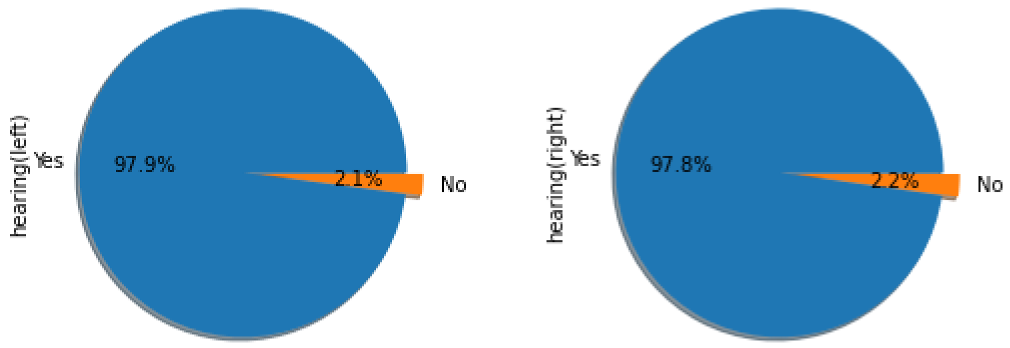
Out[34]:

	ID	smoking	gender	age	height(cm)	weight(kg)	waist(cm)	eyesight(left)	eyesight(right)
2	2	Yes	M	55	170	60	80.0	0.8	0.8
6	6	Yes	M	40	160	60	85.5	1.0	1.0
12	13	Yes	M	35	170	70	81.0	1.5	1.0
17	19	Yes	M	35	165	70	87.5	1.0	0.8
18	21	Yes	M	60	165	65	79.0	1.0	1.0

5 rows × 29 columns

In [35]:
```
plt.figure(figsize=(8,8))
plt.subplot(1,2,1)

smokeYes["hearing(left)"].value_counts().plot.pie(shadow=True,explode=[(


plt.subplot(1,2,2)
smokeYes["hearing(right)"].value_counts().plot.pie(shadow=True,explode=[

plt.tight_layout(4)
plt.show()
```

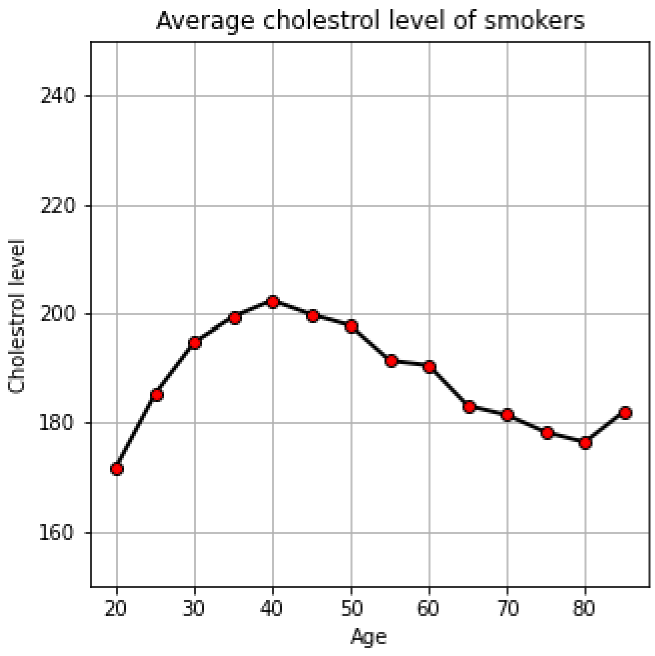
```
C:\Users\darsh\AppData\Local\Temp/ipykernel_18136/2654710606.py:10: Matplot
libDeprecationWarning: Passing the pad parameter of tight_layout() position
ally is deprecated since Matplotlib 3.3; the parameter will become keyword-
only two minor releases later.
  plt.tight_layout(4)
```



Average cholestrol level of smokers

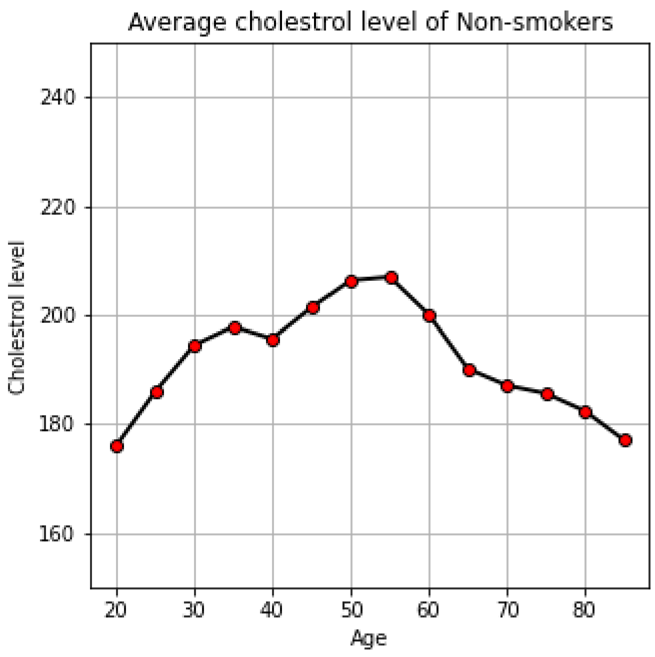
In [36]:

```

smokeYes.groupby('age').mean()['Cholesterol'].plot(ylim=(150,250),color=
                                                markerfacecolor='red',
plt.xlabel('Age')
plt.ylabel("Cholestrol level")
plt.title("Average cholestrol level of smokers")
plt.grid()
plt.show()


```



Average cholestrol level of non-smokers

In [37]:
```
smokeNo.groupby('age').mean()['Cholesterol'].plot(ylim=(150,250),color=
                                                    markerfacecolor='red'

plt.xlabel('Age')
plt.ylabel("Cholestrol level")
plt.title("Average cholestrol level of Non-smokers")
plt.grid()


plt.show()
```



In [38]:
```
smokeyes=df[df["smoking"]=="Yes"]
smokeyes.head(2)
```

Out[38]:

	ID	smoking	gender	age	height(cm)	weight(kg)	waist(cm)	eyesight(left)	eyesight(right)
2	2	Yes	M	55	170	60	80.0	0.8	0.8
6	6	Yes	M	40	160	60	85.5	1.0	1.0

2 rows × 29 columns

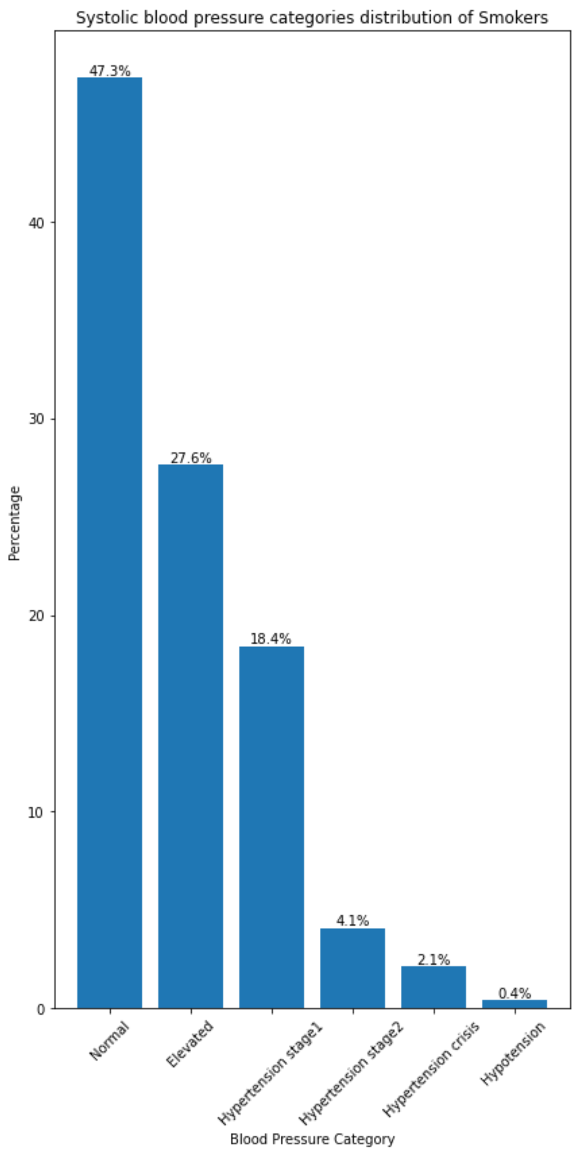
In [39]:
```
smokeno=df[df["smoking"]=="No"]
smokeno.head(2)
```

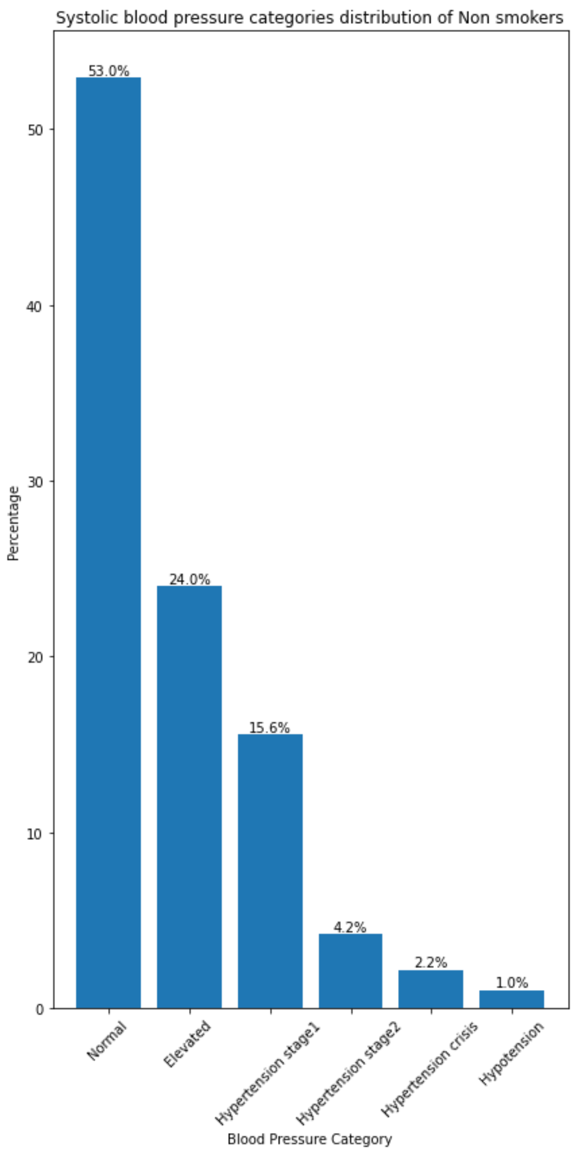
Out[39]:

	ID	smoking	gender	age	height(cm)	weight(kg)	waist(cm)	eyesight(left)	eyesight(right)
0	0	No	F	40	155	60	81.3	1.2	1.0
1	1	No	F	40	160	60	81.0	0.8	0.6

2 rows × 29 columns

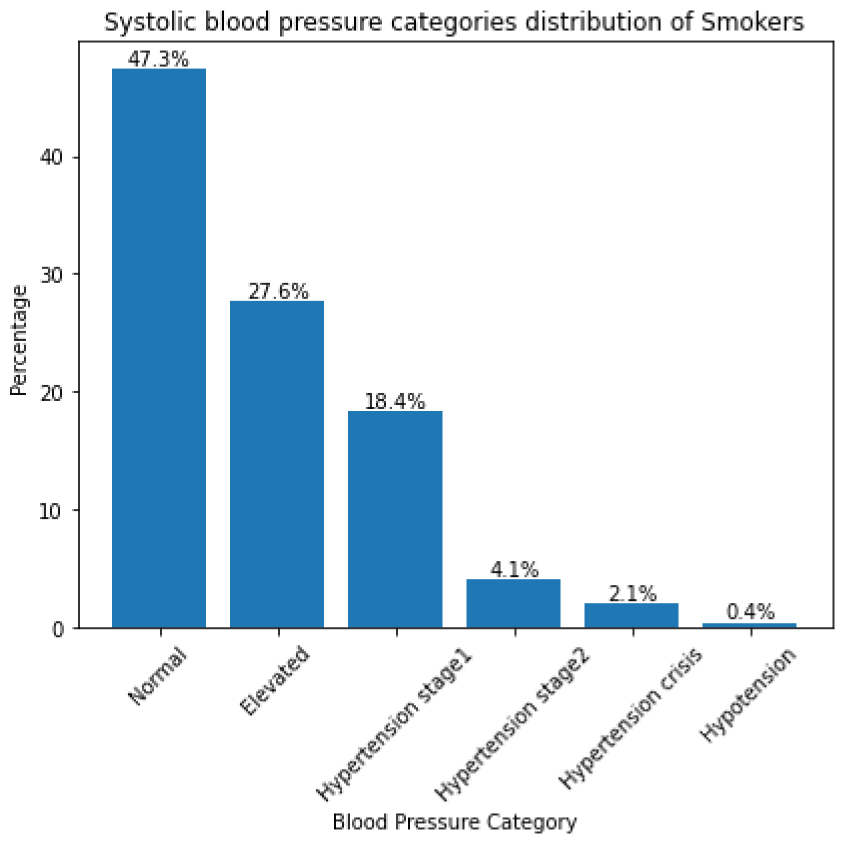
In [40]:
```python
plt.figure(figsize=(12,12))

counts = smokeyes['systolic_category'].value_counts()
counts1= smokeno['systolic_category'].value_counts()

percentages = (counts / counts.sum()) * 100
percentages1 = (counts1 / counts1.sum()) * 100

sorted_categories = percentages.sort_values(ascending=False).index
sorted_categories1 = percentages1.sort_values(ascending=False).index


plt.subplot(1,2,1)

plt.bar(sorted_categories, percentages[sorted_categories])

plt.xlabel('Blood Pressure Category')
plt.ylabel('Percentage')
plt.title('Systolic blood pressure categories distribution of Smokers')
plt.xticks(rotation=45)

for i, v in enumerate(percentages[sorted_categories]):
    plt.text(i, v, f'{v:.1f}%', ha='center', va='bottom')



plt.subplot(1,2,2)

plt.bar(sorted_categories1, percentages1[sorted_categories1])

plt.xlabel('Blood Pressure Category')
plt.ylabel('Percentage')
plt.title('Systolic blood pressure categories distribution of Non smoker
plt.xticks(rotation=45)

for i, v in enumerate(percentages1[sorted_categories1]):
    plt.text(i, v, f'{v:.1f}%', ha='center', va='bottom')

plt.tight_layout()
plt.show()

```





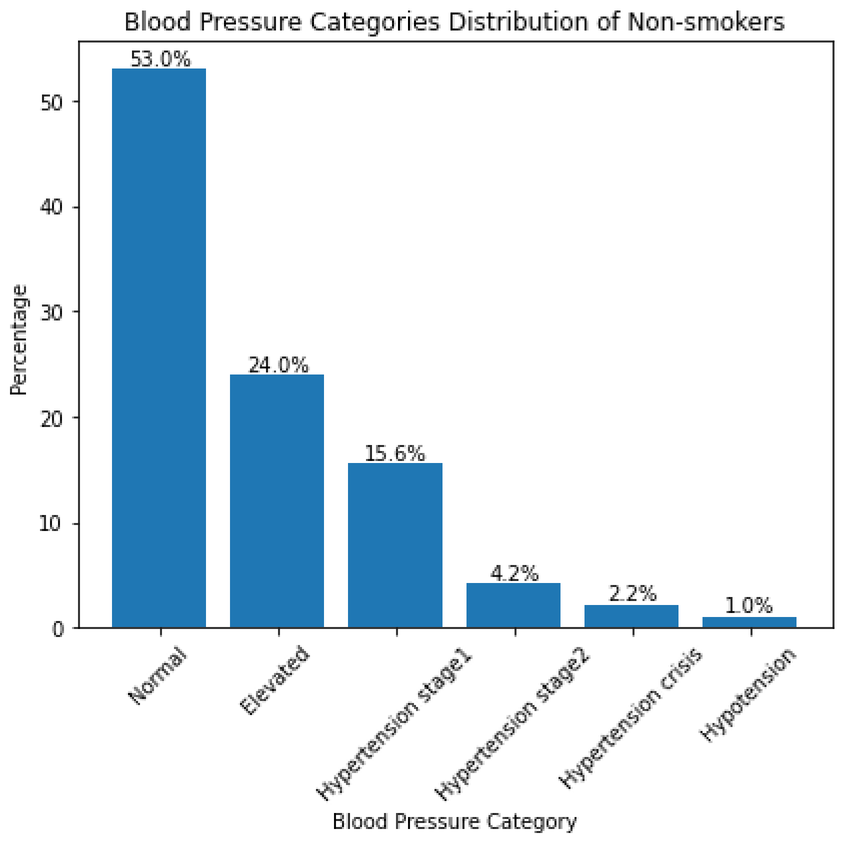
Analysis of Systoli Blood Pressure Categories Distribution of Smokers

In [41]:
```python

counts = smokeyes['systolic_category'].value_counts()

percentages = (counts / counts.sum()) * 100

sorted_categories = percentages.sort_values(ascending=False).index

plt.figure(figsize=(6,6))
plt.bar(sorted_categories, percentages[sorted_categories])

plt.xlabel('Blood Pressure Category')
plt.ylabel('Percentage')
plt.title('Systolic blood pressure categories distribution of Smokers')

plt.xticks(rotation=45)

for i, v in enumerate(percentages[sorted_categories]):
    plt.text(i, v, f'{v:.1f}%', ha='center', va='bottom')

plt.tight_layout()
plt.show()
```

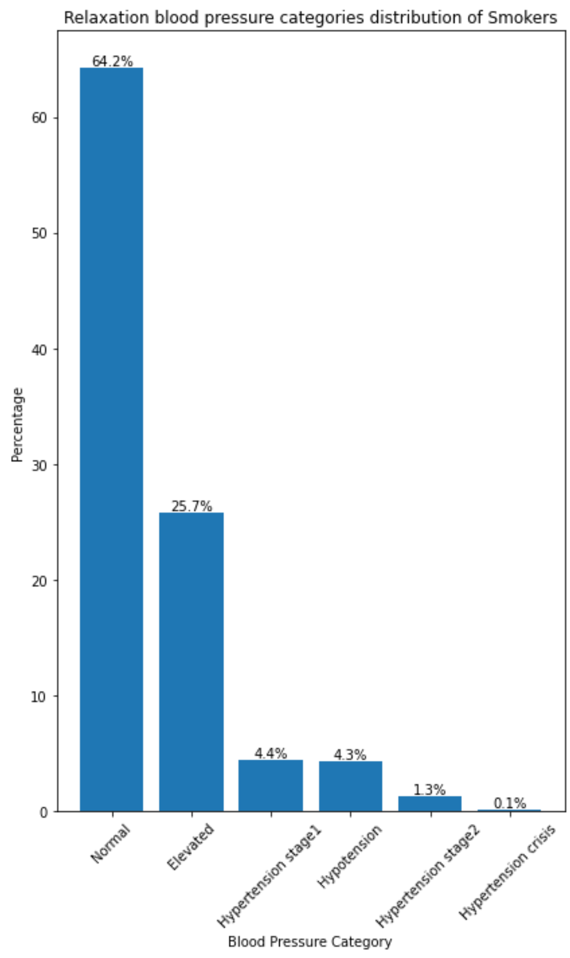


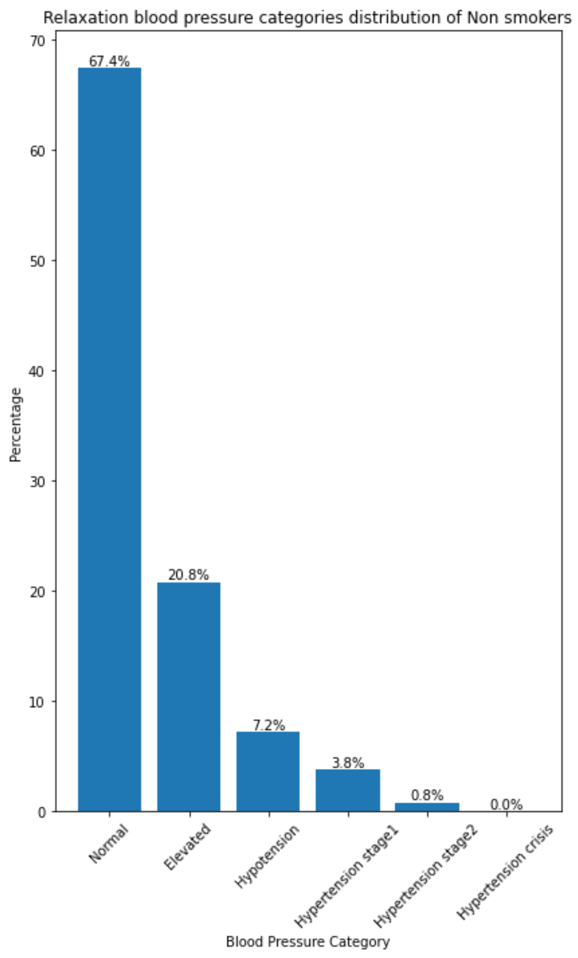
Analysis of Systolic Blood Pressure Categories Distribution of Non-smokers

In [42]:
```python

counts = smokeno['systolic_category'].value_counts()

percentages = (counts / counts.sum()) * 100

sorted_categories = percentages.sort_values(ascending=False).index

plt.figure(figsize=(6,6))
plt.bar(sorted_categories, percentages[sorted_categories])

plt.xlabel('Blood Pressure Category')
plt.ylabel('Percentage')
plt.title('Blood Pressure Categories Distribution of Non-smokers')

plt.xticks(rotation=45)

for i, v in enumerate(percentages[sorted_categories]):
    plt.text(i, v, f'{v:.1f}%', ha='center', va='bottom')

plt.tight_layout()
plt.show()
```



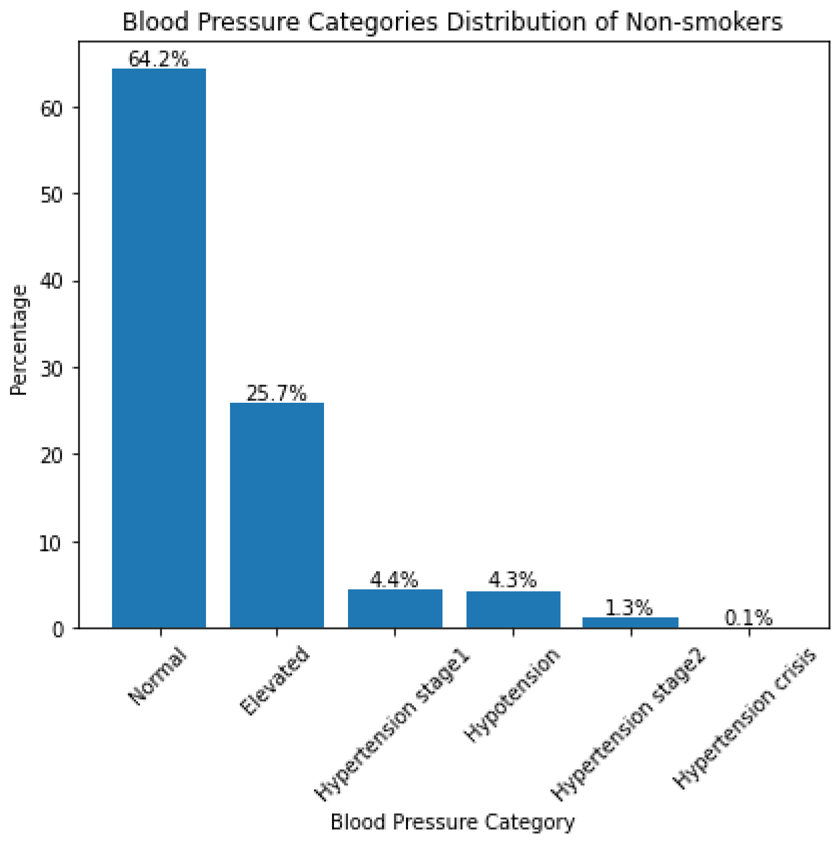
In [43]:
```python
plt.figure(figsize=(12,10))

counts = smokeyes['relaxation_category'].value_counts()
counts1= smokeno['relaxation_category'].value_counts()

percentages = (counts / counts.sum()) * 100
percentages1 = (counts1 / counts1.sum()) * 100

sorted_categories = percentages.sort_values(ascending=False).index
sorted_categories1 = percentages1.sort_values(ascending=False).index


plt.subplot(1,2,1)

plt.bar(sorted_categories, percentages[sorted_categories])

plt.xlabel('Blood Pressure Category')
plt.ylabel('Percentage')
plt.title('Relaxation blood pressure categories distribution of Smokers
plt.xticks(rotation=45)

for i, v in enumerate(percentages[sorted_categories]):
    plt.text(i, v, f'{v:.1f}%', ha='center', va='bottom')



plt.subplot(1,2,2)

plt.bar(sorted_categories1, percentages1[sorted_categories1])

plt.xlabel('Blood Pressure Category')
plt.ylabel('Percentage')
plt.title('Relaxation blood pressure categories distribution of Non smok
plt.xticks(rotation=45)

for i, v in enumerate(percentages1[sorted_categories1]):
    plt.text(i, v, f'{v:.1f}%', ha='center', va='bottom')

plt.tight_layout()
plt.show()
```





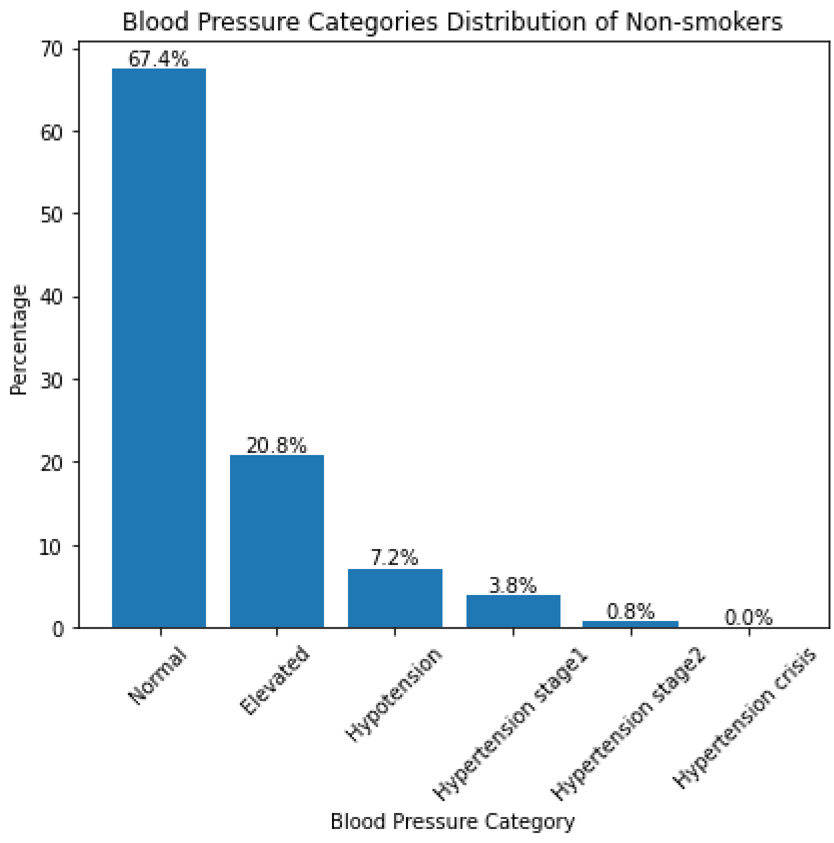
Analysis of Relaxation Blood Pressure Categories Distribution of smokers

In [44]:
```python

counts = smokeyes['relaxation_category'].value_counts()

percentages = (counts / counts.sum()) * 100

sorted_categories = percentages.sort_values(ascending=False).index

plt.figure(figsize=(6,6))
plt.bar(sorted_categories, percentages[sorted_categories])

plt.xlabel('Blood Pressure Category')
plt.ylabel('Percentage')
plt.title('Blood Pressure Categories Distribution of Non-smokers')

plt.xticks(rotation=45)

for i, v in enumerate(percentages[sorted_categories]):
    plt.text(i, v, f'{v:.1f}%', ha='center', va='bottom')

plt.tight_layout()
plt.show()
```



Analysis of Relaxation Blood Pressure Categories Distribution of Non-smokers

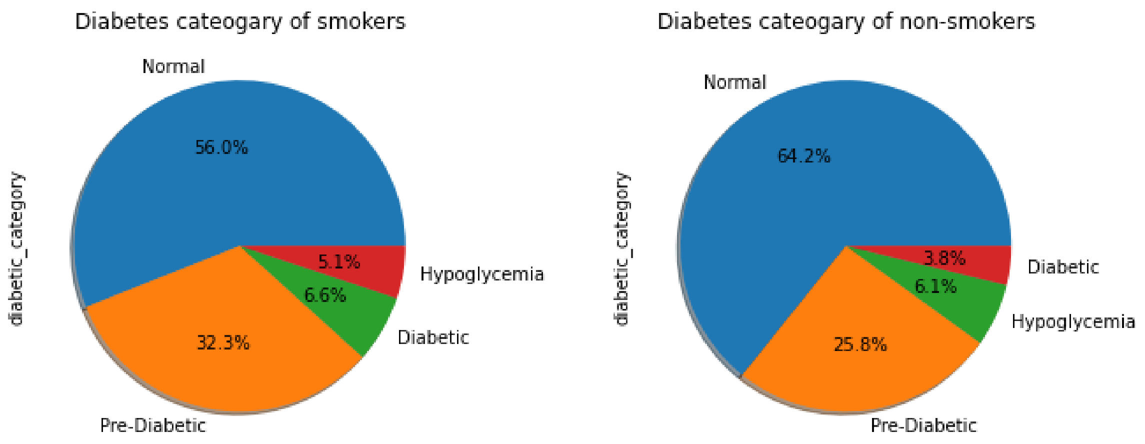
In [45]:
```python

counts = smokeno['relaxation_category'].value_counts()

percentages = (counts / counts.sum()) * 100

sorted_categories = percentages.sort_values(ascending=False).index

plt.figure(figsize=(6,6))
plt.bar(sorted_categories, percentages[sorted_categories])

plt.xlabel('Blood Pressure Category')
plt.ylabel('Percentage')
plt.title('Blood Pressure Categories Distribution of Non-smokers')

plt.xticks(rotation=45)

for i, v in enumerate(percentages[sorted_categories]):
    plt.text(i, v, f'{v:.1f}%', ha='center', va='bottom')

plt.tight_layout()
plt.show()
```



Pie chart of diabetes cateogary of smokers and non-smokers

In [46]:
```
plt.figure(figsize=(10,10))
plt.subplot(1,2,1)

smokeYes["diabetic_category"].value_counts().plot.pie(shadow=True,autop
plt.title("Diabetes cateogary of smokers")

plt.subplot(1,2,2)
smokeno["diabetic_category"].value_counts().plot.pie(shadow=True,autopct

plt.title("Diabetes cateogary of non-smokers")
plt.tight_layout(4)
plt.show()
```

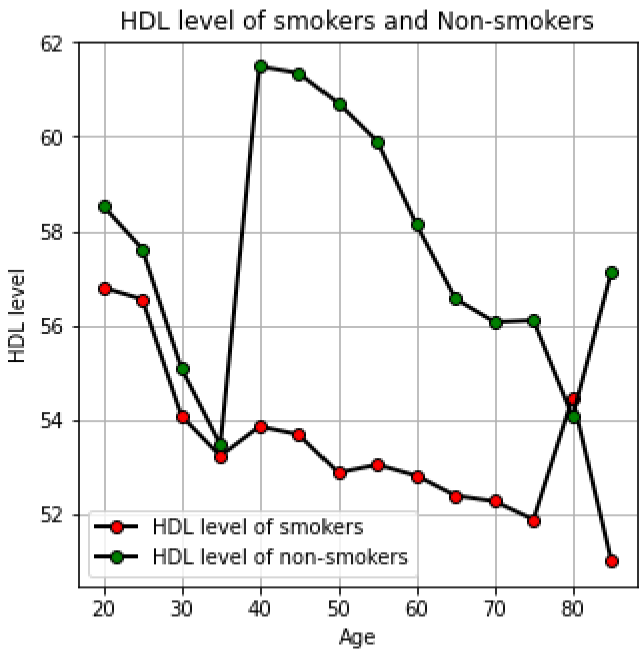
```
C:\Users\darsh\AppData\Local\Temp/ipykernel_18136/3192570241.py:11: Matplot
libDeprecationWarning: Passing the pad parameter of tight_layout() position
ally is deprecated since Matplotlib 3.3; the parameter will become keyword-
only two minor releases later.
  plt.tight_layout(4)
```



In [47]:
```
female=df[df['gender']=='F']
femaleYes=female[female['smoking']=='Yes']
femaleNo=female[female['smoking']=='No']

```

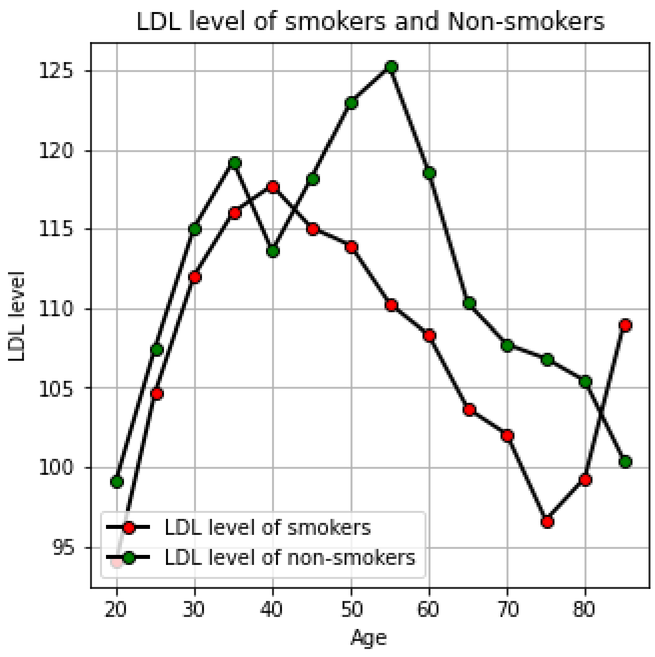
Analysis of average HDL level of smoker and non-smoker different age

In [48]:
```
plt.figure(figsize=(10,10))
smokeYes.groupby('age').mean()['HDL'].plot(color='k',marker='o',markerfa
                                              linestyle='-',linewidth=
smokeNo.groupby('age').mean()['HDL'].plot(color='k',marker='o',markerfac
                                              linestyle='-',linewidth=
plt.xlabel('Age')
plt.ylabel("HDL level")
plt.title("HDL level of smokers and Non-smokers")
plt.legend(loc='lower left')
plt.grid()
plt.show()


```



Analysis of average LDL level of smoker and non-smoker different age

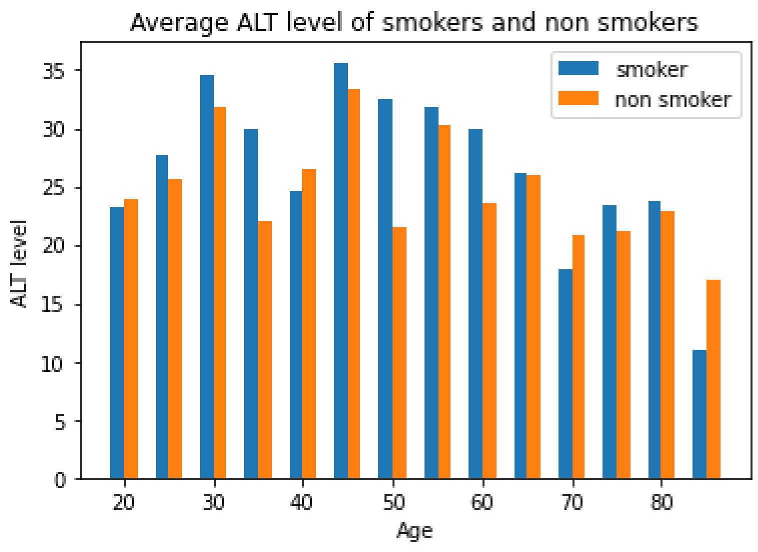
In [49]:
```
plt.figure(figsize=(10,10))
smokeYes.groupby('age').mean()['LDL'].plot(color='k',marker='o',markerfa
                                                              linestyle='-',linewidth=
smokeNo.groupby('age').mean()['LDL'].plot(color='k',marker='o',markerfac
                                                              linestyle='-',linewidth=
plt.xlabel('Age')
plt.ylabel("LDL level")
plt.title("LDL level of smokers and Non-smokers")
plt.legend(loc='lower left')
plt.grid()
plt.show()
```



Average ALT level of smokers and non smokers in different age.

In [50]:
```python
plt.figure(figsize=(10,8))

age=df['age'].unique()
altsmokers=smokeYes.groupby('age').mean()['ALT']
altnonsmokers=smokeNo.groupby('age').mean()['ALT']

fig,ax=plt.subplots()
bar_width=1.5

bar_smok_position=[x-bar_width/2 for x in age]
bar_nonsmok_position=[x+bar_width/2 for x in age]

ax.bar(bar_smok_position,altsmokers,bar_width,label='smoker')
ax.bar(bar_nonsmok_position,altnonsmokers,bar_width,label='non smoker')

ax.set_title("Average ALT level of smokers and non smokers")
ax.set_xlabel("Age")
ax.set_ylabel("ALT level")
plt.legend()

plt.show()



```

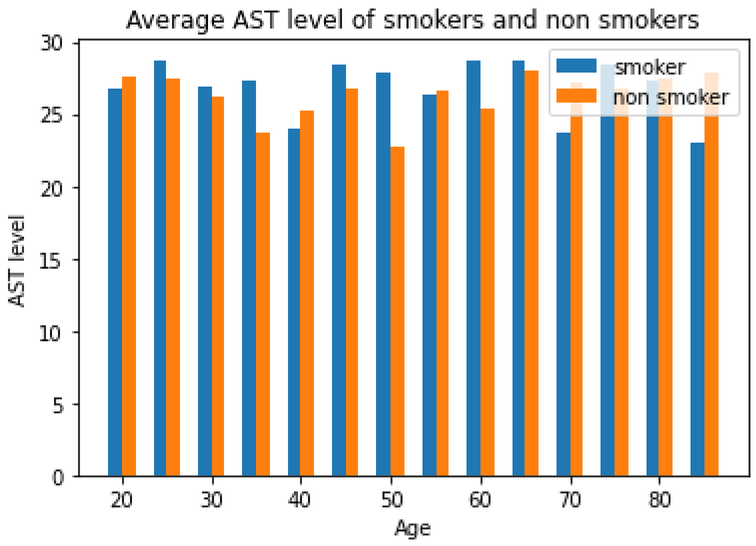
```
<Figure size 720x576 with 0 Axes>
```



Average ALT level of smokers and non smokers of different age

In [51]:
```
plt.figure(figsize=(10,10))

age=df['age'].unique()
astsmokers=smokeYes.groupby('age').mean()['AST']
astnonsmokers=smokeNo.groupby('age').mean()['AST']

fig,ax=plt.subplots()
bar_width=1.5

bar_smok_position=[x-bar_width/2 for x in age]
bar_nonsmok_position=[x+bar_width/2 for x in age]

ax.bar(bar_smok_position,astsmokers,bar_width,label='smoker')
ax.bar(bar_nonsmok_position,astnonsmokers,bar_width,label='non smoker')

ax.set_title("Average AST level of smokers and non smokers")
ax.set_xlabel("Age")
ax.set_ylabel("AST level")
plt.legend()

plt.show()

```

```
<Figure size 720x720 with 0 Axes>
```



In [ ]:
```

```

Statistical test

In [52]:
```
df.columns
```

Out[52]:
```
Index(['ID', 'smoking', 'gender', 'age', 'height(cm)', 'weight(kg)',
       'waist(cm)', 'eyesight(left)', 'eyesight(right)', 'hearing(left)',
       'hearing(right)', 'systolic', 'systolic_category', 'relaxation',
       'relaxation_category', 'fasting blood sugar', 'diabetic_category',
       'Cholesterol', 'triglyceride', 'HDL', 'LDL', 'hemoglobin',
       'serum creatinine', 'AST', 'ALT', 'Gtp', 'oral', 'dental caries',
       'tartar'],
      dtype='object')
```

To study the association between the smoking status and dental caries

H0: There is no association between the smoking status and fasting sugar level

H1:There is association between the smoking status and fasting sugar level

In [53]:
```
crosstab=pd.crosstab(df['smoking'],df['dental caries'])
crosstab
```

Out[53]:

dental caries	No	Yes
smoking		
No	6375	28862
Yes	5506	14949

In [54]:
```
import scipy.stats as stats
stats.chi2_contingency(crosstab)
```

Out[54]:
```
(600.1881467493236,
 1.5236163632659182e-132,
 1,
 array([[ 7517.25197515, 27719.74802485],
        [ 4363.74802485, 16091.25197515]]))
```

We reject null hypothesis , because p-value<0.05.Therefore there is a association between the smoking status and fasting blood sugar level.

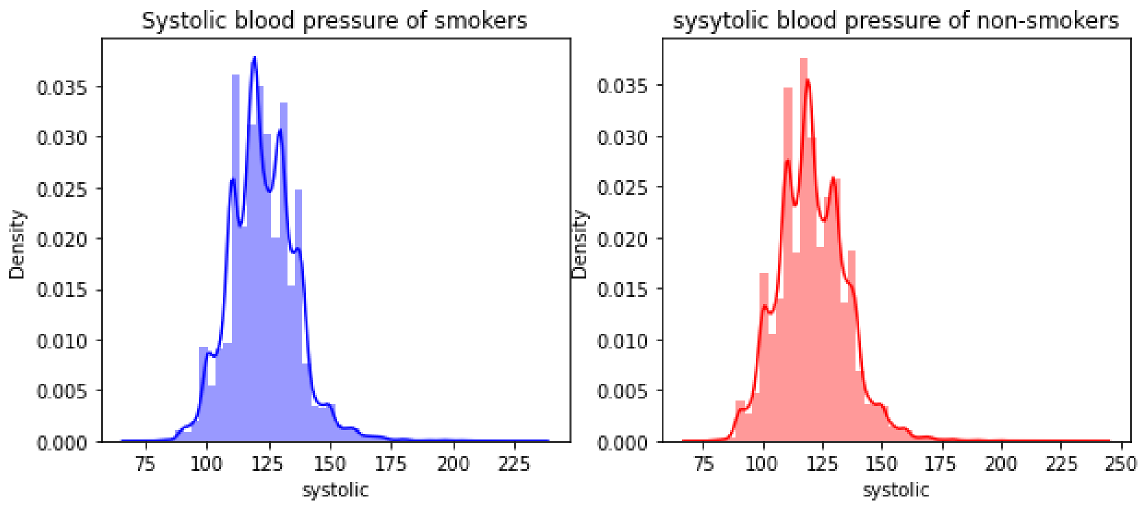
In [55]:
```
df.columns
```

Out[55]:
```
Index(['ID', 'smoking', 'gender', 'age', 'height(cm)', 'weight(kg)',
       'waist(cm)', 'eyesight(left)', 'eyesight(right)', 'hearing(left)',
       'hearing(right)', 'systolic', 'systolic_category', 'relaxation',
       'relaxation_category', 'fasting blood sugar', 'diabetic_category',
       'Cholesterol', 'triglyceride', 'HDL', 'LDL', 'hemoglobin',
       'serum creatinine', 'AST', 'ALT', 'Gtp', 'oral', 'dental caries',
       'tartar'],
      dtype='object')
```

Checking whether smoking influesing on systolic blood pressure.

In [56]:
```
#Checking for normality of systolic blood pressure

plt.figure(figsize=(10,4))
bp1=smokeyes["systolic"]
bp2=smokeno["systolic"]


plt.subplot(1,2,1)
sns.distplot(bp1,color='b')
plt.title("Systolic blood pressure of smokers")

plt.subplot(1,2,2)
sns.distplot(bp2,color='r')
plt.title("sysytolic blood pressure of non-smokers")

plt.show()
```

```
C:\Users\darsh\anaconda3\lib\site-packages\seaborn\distributions.py:2619: F
utureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-le
vel function with similar flexibility) or `histplot` (an axes-level functio
n for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\darsh\anaconda3\lib\site-packages\seaborn\distributions.py:2619: F
utureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-le
vel function with similar flexibility) or `histplot` (an axes-level functio
n for histograms).
  warnings.warn(msg, FutureWarning)
```



H0 : Smoking not influesing on systolic blood pressure of smoker

H1 : Smoking influesing on systolic blood pressure

In [57]:
```
from scipy.stats import ttest_ind # t test for relaxation bp
t3=smokeyes['systolic']
t4=smokeno['systolic']
stat,p_value=ttest_ind(t3,t4)
print("statistic= ",stat)
print("p value= ",p_value)

alpha = 0.05
if p_value < alpha:
    print("Smoking influesing on systolic blood pressure")
else:
    print("Smoking not influesing on systolic blood pressure of smoker.'


```

```
statistic=  17.299182876018282
p value=  7.140975601346086e-67
Smoking influesing on systolic blood pressure
```

In [ ]:
```

```

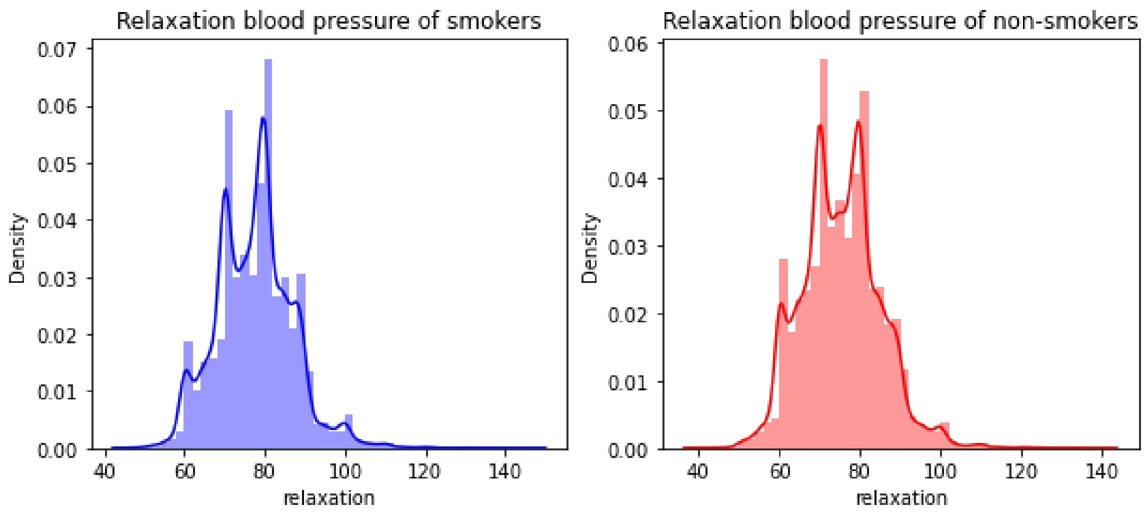
In [ ]:
```

```

Checking whether smoking influesing on relaxation blood pressure.

In [58]:

```python
#Checking for normality of relaxation blood pressure
plt.figure(figsize=(10,4))
bp3=smokeyes["relaxation"]
bp4=smokeno["relaxation"]


plt.subplot(1,2,1)
sns.distplot(bp3,color='b')
plt.title("Relaxation blood pressure of smokers")

plt.subplot(1,2,2)
sns.distplot(bp4,color='r')
plt.title("Relaxation blood pressure of non-smokers")

plt.show()
```

```
C:\Users\darsh\anaconda3\lib\site-packages\seaborn\distributions.py:2619: F
utureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-le
vel function with similar flexibility) or `histplot` (an axes-level functio
n for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\darsh\anaconda3\lib\site-packages\seaborn\distributions.py:2619: F
utureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-le
vel function with similar flexibility) or `histplot` (an axes-level functio
n for histograms).
  warnings.warn(msg, FutureWarning)
```



H0 : Smoking not influesing on relaxation blood pressure of smoker

H1 : Smoking influesing on relaxation blood pressure

In [59]:
```
from scipy.stats import ttest_ind # t test for relaxation bp
t3=smokeyes['relaxation']
t4=smokeno['relaxation']
stat,p=ttest_ind(t3,t4)
print("statistic= ",stat)
print("p value= ",p)

alpha = 0.05
if p_value < alpha:
    print("Smoking influesing on systolic blood pressure")
else:
    print("Smoking not influesing on systolic blood pressure of smoker.'

```

```
statistic=  25.71088809521144
p value=  6.221084836058103e-145
Smoking influesing on systolic blood pressure
```

In [ ]:

In [ ]:

Study the association between the smoking status and fasting blood sugar level

H0: There is no association between the smoking status and fasting sugar level

H1:There is association between the smoking status and fasting sugar level

In [60]:
```
crosstab=pd.crosstab(df['smoking'],df['diabetic_category'])
crosstab
```

Out[60]:

diabetic_category	Diabetic	Hypoglycemia	Normal	Pre-Diabetic
smoking				
No	1344	2165	22638	9090
Yes	1347	1049	11459	6600

In [61]:
```
import scipy.stats as stats
stats.chi2_contingency(crosstab)
```

Out[61]:
```
(564.0462666344756,
 6.270104551318755e-122,
 3,
 array([[ 1702.62815126,  2033.53655821, 21573.58308195,  9927.25220858],
        [  988.37184874,  1180.46344179, 12523.41691805,  5762.74779142]]))
```

In [ ]:

Checking whether smokers cholestrol level influencing systolic blood pressure and relaxation blood pressure

In [62]:
```
#using levene's method to test the whether variance of group are equal.
```

In [63]:
```
#levenes test for systolic blood pressure
from scipy.stats import levene

bp1=smokeyes['Cholesterol'][df['systolic_category']=='Hypotension']
bp2=smokeyes['Cholesterol'][df['systolic_category']=='Normal']
bp3=smokeyes['Cholesterol'][df['systolic_category']=='Elevated']
bp4=smokeyes['Cholesterol'][df['systolic_category']=='Hypertension stage
bp5=smokeyes['Cholesterol'][df['systolic_category']=='Hypertension stage
bp6=smokeyes['Cholesterol'][df['systolic_category']=='Hypertension cris


statistic, p_value = levene(bp1,bp2,bp3,bp4,bp5,bp6)


alpha = 0.05

print('Statistic :',statistic)
print('P-value : ',p_value)

if p_value < alpha:
    print("The variances are significantly different (reject the null hy
else:
    print("The variances are not significantly different (fail to reject
```

```
Statistic : 8.035912185426364
P-value :  1.3973650420756164e-07
The variances are significantly different (reject the null hypothesis of eq
ual variances).
```

In [64]:
```
# levene's test for relaxation blood pressure

bp1=smokeyes['Cholesterol'][df['relaxation_category']=='Hypotension']
bp2=smokeyes['Cholesterol'][df['relaxation_category']=='Normal']
bp3=smokeyes['Cholesterol'][df['relaxation_category']=='Elevated']
bp4=smokeyes['Cholesterol'][df['relaxation_category']=='Hypertension sta
bp5=smokeyes['Cholesterol'][df['relaxation_category']=='Hypertension sta
bp6=smokeyes['Cholesterol'][df['relaxation_category']=='Hypertension cr


statistic, p_value = levene(bp1,bp2,bp3,bp4,bp5,bp6)


alpha = 0.05

print('Statistic :',statistic)
print('P-value : ',p_value)

if p_value < alpha:
    print("The variances are significantly different (reject the null hy
else:
    print("The variances are not significantly different (fail to reject
```

```
Statistic : 4.950484091083053
P-value :  0.00015644784666161838
The variances are significantly different (reject the null hypothesis of eq
ual variances).
```

Using kruskal-wallis test checking whether smokers cholestrol level influencing systolic blood pressure

Ho: cholestrol level do not influencing systolic blood pressure

H1: cholestrol level influencing systolic blood pressure

In [65]:
```
import scipy.stats as stats

bp1=smokeyes['Cholesterol'][df['systolic_category']=='Hypotension']
bp2=smokeyes['Cholesterol'][df['systolic_category']=='Normal']
bp3=smokeyes['Cholesterol'][df['systolic_category']=='Elevated']
bp4=smokeyes['Cholesterol'][df['systolic_category']=='Hypertension stage
bp5=smokeyes['Cholesterol'][df['systolic_category']=='Hypertension stage
bp6=smokeyes['Cholesterol'][df['systolic_category']=='Hypertension cris

# Perform the Kruskal-Wallis test
statistic, p_value = stats.kruskal(bp1,bp2,bp3,bp4,bp5,bp6)


print("Kruskal-Wallis Test")
print("Statistic:", statistic)
print("P-value:", p_value)

# Check the significance level and draw conclusions
alpha = 0.05
if p_value < alpha:
    print("There is a significant difference between the groups.")
else:
    print("There is no significant difference between the groups.")
```

```
Kruskal-Wallis Test
Statistic: 112.85332939360227
P-value: 1.0216312842082404e-22
There is a significant difference between the groups.
```

In [ ]:
```

```

In [ ]:
```

```

Checking whether smokers cholestrol level influencing relaxation blood pressure

Ho: cholestrol level do not influencing systolic blood pressure

H1: cholestrol level influencing systolic blood pressure

In [66]:
```

bp1=smokeyes['Cholesterol'][df['relaxation_category']=='Hypotension']
bp2=smokeyes['Cholesterol'][df['relaxation_category']=='Normal']
bp3=smokeyes['Cholesterol'][df['relaxation_category']=='Elevated']
bp4=smokeyes['Cholesterol'][df['relaxation_category']=='Hypertension sta
bp5=smokeyes['Cholesterol'][df['relaxation_category']=='Hypertension sta
bp6=smokeyes['Cholesterol'][df['relaxation_category']=='Hypertension cri

# Perform the Kruskal-Wallis test
statistic, p_value = stats.kruskal(bp1,bp2,bp3,bp4,bp5,bp6)


print("Kruskal-Wallis Test")
print("Statistic:", statistic)
print("P-value:", p_value)

# Check the significance level and draw conclusions
alpha = 0.05
if p_value < alpha:
    print("There is a significant difference between the groups.")
else:
    print("There is no significant difference between the groups.")
```

```
Kruskal-Wallis Test
Statistic: 210.87431690442494
P-value: 1.3373403109815881e-43
There is a significant difference between the groups.
```

In [67]:
```
df['diabetic_category'].unique()
```

Out[67]:
```
array(['Normal', 'Pre-Diabetic', 'Hypoglycemia', 'Diabetic'], dtype=object)
```

Checking whether smokers diabetic level influencing on serum creatinine

In [68]:
```
# Levene's test for relaxation blood pressure

from scipy.stats import levene

sc1=smokeyes['serum creatinine'][df['diabetic_category']=='Hypoglycemia
sc2=smokeyes['serum creatinine'][df['diabetic_category']=='Normal']
sc3=smokeyes['serum creatinine'][df['diabetic_category']=='Pre-Diabetic
sc4=smokeyes['serum creatinine'][df['diabetic_category']=='Diabetic']



statistic, p_value = levene(sc1,sc2,sc3,sc4)


alpha = 0.05

print('Statistic :',statistic)
print('P-value : ',p_value)

if p_value < alpha:
    print("The variances are significantly different (reject the null hy
else:
    print("The variances are not significantly different (fail to reject
```

```
Statistic : 9.784655477556692
P-value :  1.9055390702653126e-06
The variances are significantly different (reject the null hypothesis of eq
ual variances).
```

Kruskal-Wallis Test for checking whether smokers diabetic level influencing on serum creatine.

In [69]:
```
from scipy.stats import kruskal

sc1=smokeyes['serum creatinine'][df['diabetic_category']=='Hypoglycemia
sc2=smokeyes['serum creatinine'][df['diabetic_category']=='Normal']
sc3=smokeyes['serum creatinine'][df['diabetic_category']=='Pre-Diabetic
sc4=smokeyes['serum creatinine'][df['diabetic_category']=='Diabetic']



statistic, p_value = kruskal(sc1,sc2,sc3,sc4)

print("Kruskal-Wallis Test")
print("Statistic:", statistic)
print("P-value:", p_value)

# Check the significance level and draw conclusions
alpha = 0.05
if p_value < alpha:
    print("There is a significant difference between serum creatinine of
else:
    print("There is no significant difference between serum creatinine c


```

```
Kruskal-Wallis Test
Statistic: 17.37610751015689
P-value: 0.0005913810964908272
There is a significant difference between serum creatinine of smokers  .
```

In [ ]:
```

```

In [ ]:
```

```

In [70]:
```
#Model building
```

In [71]:
```
df.head(3)
```

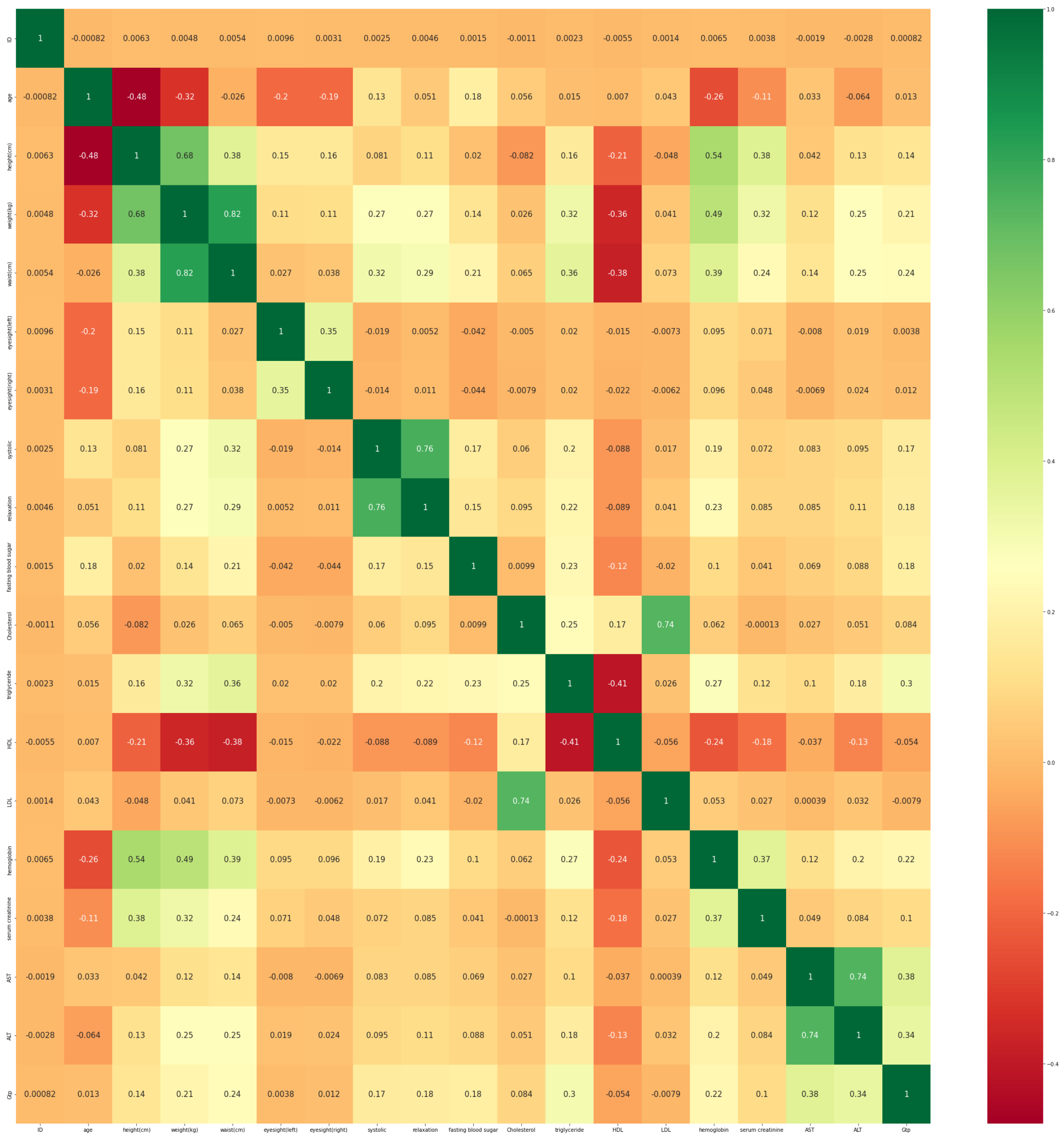
Out[71]:

	ID	smoking	gender	age	height(cm)	weight(kg)	waist(cm)	eyesight(left)	eyesight(right)
0	0	No	F	40	155	60	81.3	1.2	1.0
1	1	No	F	40	160	60	81.0	0.8	0.6
2	2	Yes	M	55	170	60	80.0	0.8	0.8

3 rows × 29 columns

In [72]:
```
plt.figure(figsize=(40,40))
sns.heatmap(df.corr(),annot=True,cmap='RdYlGn',annot_kws={'size':15})
```

Out[72]:
```
<AxesSubplot:>
```



In [73]:
```
from sklearn.metrics import classification_report,PrecisionRecallDisplay
from sklearn.metrics import precision_score,recall_score,f1_score
```

In [74]:
```
from sklearn.preprocessing import LabelEncoder
for column in df.columns:
    if df[column].dtype==np.number:
        continue
    else:
        df[column]=LabelEncoder().fit_transform(df[column])
```

```
C:\Users\darsh\AppData\Local\Temp/ipykernel_18136/4259945406.py:3: Deprecat
ionWarning: Converting `np.inexact` or `np.floating` to a dtype is deprecat
ed. The current result is `float64` which is not strictly correct.
  if df[column].dtype==np.number:
```

In [75]:
```
df.columns
```

Out[75]:
```
Index(['ID', 'smoking', 'gender', 'age', 'height(cm)', 'weight(kg)',
       'waist(cm)', 'eyesight(left)', 'eyesight(right)', 'hearing(left)',
       'hearing(right)', 'systolic', 'systolic_category', 'relaxation',
       'relaxation_category', 'fasting blood sugar', 'diabetic_category',
       'Cholesterol', 'triglyceride', 'HDL', 'LDL', 'hemoglobin',
       'serum creatinine', 'AST', 'ALT', 'Gtp', 'oral', 'dental caries',
       'tartar'],
      dtype='object')
```

In [76]:
```
x=df.drop(columns=['ID','eyesight(left)','eyesight(right)','hearing(left
y=df['diabetic_category']
```

In [77]:
```
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.20,random_sta
```

LogisticRegression

In [78]:
```
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
model.fit(xtrain,ytrain)
```

```
C:\Users\darsh\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.p
y:763: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sci
kit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regr
ession (https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression)
  n_iter_i = _check_optimize_result(
```

Out[78]:
```
LogisticRegression()
```

In [79]:
```
ypred1=model.predict(xtest)
```

In [80]:
```
from sklearn.metrics import accuracy_score
print("Test Accuracy: ",accuracy_score(ytest,ypred1))
print("Precision: ",precision_score(ytest,ypred1,average='weighted'))
```

```
Test Accuracy:  0.625100996498788
Precision:  0.5706326704136191
```

In [81]:
```
ypredd=model.predict(xtrain)
```

In [82]:
```
from sklearn.metrics import accuracy_score
print("Train Accuracy: ",accuracy_score(ytrain,ypredd))
```

```
Train Accuracy:  0.6209682849639755
```

In [83]:
```python
from sklearn.metrics import classification_report
print(classification_report(ytest,ypred1))
```

```
              precision    recall  f1-score   support

           0       0.27      0.01      0.02       493
           1       0.50      0.00      0.00       653
           2       0.65      0.92      0.76      6869
           3       0.46      0.20      0.28      3124

    accuracy                           0.63     11139
   macro avg       0.47      0.28      0.26     11139
weighted avg       0.57      0.63      0.55     11139
```

In [84]:
```python
print()
```

KNN

In [85]:
```python
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import f1_score
```

In [86]:
```python
clf=KNeighborsClassifier(n_neighbors=5)
clf.fit(xtrain,ytrain)
ypred2=clf.predict(xtest)
```

In [87]:
```python
from sklearn.metrics import accuracy_score
print("Test Accuracy: ",accuracy_score(ytest,ypred2))
```

```
Test Accuracy:  0.5819193823503007
```

In [88]:
```python
ypredd2=clf.predict(xtrain)
```

In [89]:
```python
from sklearn.metrics import accuracy_score
print("Train Accuracy: ",accuracy_score(ytrain,ypredd2))
```

```
Train Accuracy:  0.7083249163917132
```

In [90]:
```python
print(classification_report(ytest,ypred2))
```

```
              precision    recall  f1-score   support

           0       0.18      0.14      0.15       493
           1       0.14      0.08      0.10       653
           2       0.66      0.79      0.72      6869
           3       0.43      0.29      0.35      3124

    accuracy                           0.58     11139
   macro avg       0.35      0.32      0.33     11139
weighted avg       0.54      0.58      0.55     11139
```

In [ ]:

```

```

In [91]:

```python
from sklearn.neighbors import KNeighborsClassifier
def elbow(k):
    test_error=[]
    for i in k:
        clf=KNeighborsClassifier(n_neighbors=i)
        clf.fit(xtrain,ytrain)
        tmp=clf.predict(xtest)
        tmp=f1_score(ytest,tmp,average='micro')
        error=1-tmp
        test_error.append(error)
    return test_error
```

In [92]:

```python
k=range(6,20)
test=elbow(k)
print(test)
```

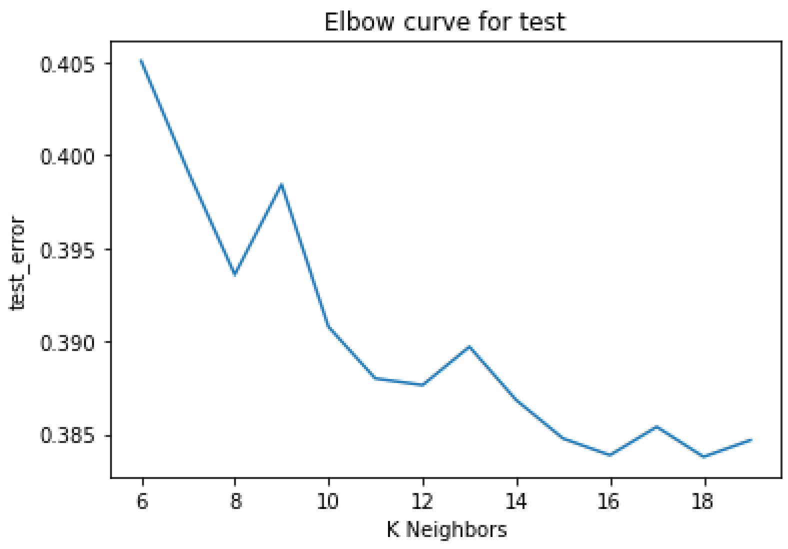
```
[0.40506329113924056, 0.399138163210342, 0.3935721339438011, 0.398419965885
6271, 0.3907891193105306, 0.38800610467726004, 0.38764700601490254, 0.38971
18233234581, 0.3868390340245983, 0.3847742167160427, 0.383876470060149, 0.3
854026393751683, 0.3837866953945597, 0.3846844420504534]
```

In [93]:

```python
plt.plot(k,test)
plt.xlabel("K Neighbors")
plt.ylabel("test_error")
plt.title("Elbow curve for test")
```

Out[93]:

```
Text(0.5, 1.0, 'Elbow curve for test')
```



In [94]:

```python
knn=KNeighborsClassifier(n_neighbors=16)
knn.fit(xtrain,ytrain)
```

Out[94]:

```
KNeighborsClassifier(n_neighbors=16)
```

In [95]:

```python
ytrain_prdct=knn.predict(xtrain)
ytest_prdct=knn.predict(xtest)
```

In [96]:
```
print("Test Accuracy :",accuracy_score(ytest_prdct,ytest))
print("Train Accuracy :",accuracy_score(ytrain_prdct,ytrain))
```
```
Test Accuracy : 0.616123529939851
Train Accuracy : 0.6430094494197922
```

In [ ]:
```

```

Naive Bayesian Classification

In [97]:
```
from sklearn.naive_bayes import GaussianNB
gnb=GaussianNB()
gnb.fit(xtrain,ytrain)
```
Out[97]:
```
GaussianNB()
```

In [98]:
```
ypred3=gnb.predict(xtest)
ypredd3=gnb.predict(xtrain)
```

In [99]:
```
from sklearn.metrics import accuracy_score
print("Test Accuracy: ",accuracy_score(ytest,ypred3))
print("Train Accuracy: ",accuracy_score(ytrain,ypredd3))
```
```
Test Accuracy:  0.5688122811742526
Train Accuracy:  0.5723969205216259
```

In [100]:
```
print(classification_report(ytest,ypred3))
```
```
              precision    recall  f1-score   support

           0       0.17      0.21      0.19       493
           1       0.10      0.03      0.04       653
           2       0.67      0.77      0.71      6869
           3       0.39      0.31      0.34      3124

    accuracy                           0.57     11139
   macro avg       0.33      0.33      0.32     11139
weighted avg       0.53      0.57      0.55     11139
```

RandomForest and DecisionTreeClassifier

In [101]:
```
from sklearn.tree import DecisionTreeClassifier
dtree=DecisionTreeClassifier()
dtree.fit(xtrain,ytrain)
```
Out[101]:
```
DecisionTreeClassifier()
```

In [102]:
```
ypred4=dtree.predict(xtest)
ypredd4=dtree.predict(xtrain)
```

In [103]:
```python
print("Test Accuracy: ",accuracy_score(ytest,ypred4))
print("Train Accuracy: ",accuracy_score(ytrain,ypredd4))
```
```
Test Accuracy:  0.6531106921626717
Train Accuracy:  1.0
```

In [104]:
```python
print(classification_report(ytest,ypred4))
```
```
              precision    recall  f1-score   support

           0       0.35      0.42      0.38       493
           1       0.35      0.37      0.36       653
           2       0.76      0.75      0.75      6869
           3       0.54      0.55      0.54      3124

    accuracy                           0.65     11139
   macro avg       0.50      0.52      0.51     11139
weighted avg       0.66      0.65      0.66     11139
```

Random Forest

In [105]:
```python
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier()
rf.fit(xtrain,ytrain)
```
Out[105]:
```
RandomForestClassifier()
```

In [106]:
```python
ypred5=rf.predict(xtest)
ypredd5=rf.predict(xtrain)
```

In [107]:
```python
print("Test Accuracy: ",accuracy_score(ytest,ypred5))
print("Train Accuracy: ",accuracy_score(ytrain,ypredd5))

```
```
Test Accuracy:  0.7467456683723853
Train Accuracy:  1.0
```

In [108]:
```python
# hyperparameter tuning

```

GridSearchCV

In [110]:
```python
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import RandomizedSearchCV
```

In [111]:
```python
param_grid = {
    'n_estimators': [25, 50, 100, 150],
    'max_features': ['sqrt', 'log2', None],
    'max_depth': [3, 6, 9],
    'max_leaf_nodes': [3, 6, 9],
}
```

In [113]:
```
grid_search = GridSearchCV(RandomForestClassifier(),
                           param_grid=param_grid)
grid_search.fit(xtrain, ytrain)
print(grid_search.best_estimator_)
```

```
RandomForestClassifier(max_depth=9, max_features=None, max_leaf_nodes=9,
                       n_estimators=150)
```

In [114]:
```
model_grid = RandomForestClassifier(max_depth=9,
                                    max_features=None,
                                    max_leaf_nodes=9,
                                    n_estimators=150)
model_grid.fit(xtrain, ytrain)
y_pred_grid = model.predict(xtest)
print(classification_report(y_pred_grid, ytest))
```

```
              precision    recall  f1-score   support

           0       0.01      0.27      0.02        15
           1       0.00      0.50      0.00         2
           2       0.92      0.65      0.76      9770
           3       0.20      0.46      0.28      1352

    accuracy                           0.63     11139
   macro avg       0.28      0.47      0.26     11139
weighted avg       0.83      0.63      0.70     11139
```

In [116]:
```
print("Test Accuracy: ",accuracy_score(y_pred_grid, ytest))
```

```
Test Accuracy:  0.625100996498788
```

In [118]:
```
ytrain_pred_grid = model.predict(xtrain)
print("Train Accuracy",accuracy_score(ytrain_pred_grid, ytrain))
```

```
Train Accuracy 0.6209682849639755
```

In [119]:
```
# Randomizedsearch
```

In [125]:
```
param_grid = {
    'n_estimators': [25, 50, 100, 150],
    'max_features': ['sqrt', 'log2', None],
    'max_depth': [9, 12, 15],
    'max_leaf_nodes': [9, 12, 15],
}
```

In [126]:
```
random_search = RandomizedSearchCV(RandomForestClassifier(),
                                   param_grid)
random_search.fit(xtrain, ytrain)
print(random_search.best_estimator_)
```

```
RandomForestClassifier(max_depth=15, max_features=None, max_leaf_nodes=15,
                       n_estimators=25)
```

In [128]:
```
model_random = RandomForestClassifier(max_depth=15,
                                      max_features=None,
                                      max_leaf_nodes=15,
                                      n_estimators=25)
model_random.fit(xtrain, ytrain)
ytest_pred_rand1 = model.predict(xtest)
ytrain_pred_rand1=model.predict(xtrain)
print("Test Accuracy :",accuracy_score(ytest_pred_rand1, ytest))
print("Train Accuracy :",accuracy_score(ytrain_pred_rand1,ytrain))
```

```
Test Accuracy : 0.625100996498788
Train Accuracy : 0.6209682849639755
```

Xgboost

In [ ]:
```
params={
    'objective':'multi:softmax',
    'num_class':len(set(y)),
    'max_depth':3,
    'learning_rate':0.1,
    'n_estimators':100
}
```

In [134]:
```
!pip install xgboost
```

```
Collecting xgboost
  Downloading xgboost-1.7.6-py3-none-win_amd64.whl (70.9 MB)
Requirement already satisfied: numpy in c:\users\darsh\anaconda3\lib\site-p
ackages (from xgboost) (1.20.3)
Requirement already satisfied: scipy in c:\users\darsh\anaconda3\lib\site-p
ackages (from xgboost) (1.7.1)
Installing collected packages: xgboost
Successfully installed xgboost-1.7.6
```

In [135]:
```
import xgboost as xgb

```

In [137]:
```
model=xgb.XGBClassifier()

model.fit(xtrain,ytrain)
ytest_pred=model.predict(xtest)
ytrain_pred=model.predict(xtrain)

print("Test ACcuracy: ",accuracy_score(ytest_pred,ytest))
print("Train Accuracy: ",accuracy_score(ytrain_pred,ytrain))
```

```
Test ACcuracy:  0.6548164108088698
Train Accuracy:  0.750723856979328
```

In [ ]:
```

```

In [138]:
```
booster=['gbtree','gblinear']
base_score=[0.25,0.5,0.75,1]
n_estimators = [100, 500, 900, 1100, 1500]
max_depth = [2, 3, 5, 10, 15]
booster=['gbtree','gblinear']
learning_rate=[0.05,0.1,0.15,0.20]
min_child_weight=[1,2,3,4]

# Define the grid of hyperparameters to search
hyperparameter_grid = {
    'n_estimators': n_estimators,
    'max_depth':max_depth,
    'learning_rate':learning_rate,
    'min_child_weight':min_child_weight,
    'booster':booster,
    'base_score':base_score
    }
```

In [140]:
```
random_cv = RandomizedSearchCV(estimator=xgb.XGBClassifier(),
            param_distributions=hyperparameter_grid,
            cv=5, n_iter=50,
            scoring = 'neg_mean_absolute_error',n_jobs = 4,
            verbose = 5,
            return_train_score = True,
            random_state=42)
```

In [ ]:
```
random_cv.fit(xtrain,ytrain)

random_cv.best_estimator_

```

In [ ]:
```

```